SAND REPORT
SAND2003-3820
Unlimited Release
Printed October 2003

Algorithm Development for Prognostics and Health Management (PHM)

Laura P. Swiler, James E. Campbell, Kelly S. Lowder, and Adele B. Doser

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, A Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States
Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (856)576-5728
E-Mail reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail orders@ntis.fedworld.gov
Online order: htt;://www.ntis.gov/ordering.htm

SAND 2003-3820
Unlimited Release
Printed October 2003

Algorithm Development for Prognostics and Health Management (PHM)

Laura P. Swiler, James E. Campbell, Kelly S. Lowder, and Adele B. Doser
Systems Reliability Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1176

Abstract

This report summarizes the results of a three-year LDRD project on prognostics and health management. "Prognostics" refers to the capability to predict the probability of system failure over some future time interval (an alternative definition is the capability to predict the remaining useful life of a system). Prognostics are integrated with health monitoring (through inspections, sensors, etc.) to provide an overall PHM capability that optimizes maintenance actions and results in higher availability at a lower cost. Our goal in this research was to develop PHM tools that could be applied to a wide variety of equipment (repairable, non-repairable, manufacturing, weapons, battlefield equipment, etc.) and require minimal customization to move from one system to the next. Thus, our approach was to develop a toolkit of reusable software objects/components and architecture for their use.

We have developed two software tools: an Evidence Engine and a Consequence Engine. The Evidence Engine integrates information from a variety of sources in order to take into account all the evidence that impacts a prognosis for system health. The Evidence Engine has the capability for feature extraction, trend detection, information fusion through Bayesian Belief Networks (BBN), and estimation of remaining useful life. The Consequence Engine involves algorithms to analyze the consequences of various maintenance actions. The Consequence Engine takes as input a maintenance and use schedule, spares information, and time-to-failure data on components, then generates maintenance and failure events, and evaluates performance measures such as equipment availability, mission capable rate, time to failure, and cost.

This report summarizes the capabilities we have developed, describes the approach and architecture of the two engines, and provides examples of their use.

Table of Contents

Abstract 3
Table of Contents 5
Table of Figures 7
1. Introduction 9
1.1. Motivation for PHM 9
1.2. PHM Functional Architecture 10
1.3. PHM Technology Needs 11
1.4. LDRD Focus 12
2. Evidence Engine 14
2.1. Bayesian Belief Networks 14
2.1.1. Bayesian Belief Networks - Background 14
2.1.2. Bayesian Belief Network – Example 15
2.2. Self-Organizing Maps 26
2.2.1. Background on Self Organizing Maps 26
2.2.2. SOM Analysis of Available Data 26
2.2.3. SOM Analysis of Simulated Faulty Data 29
2.3. Vibration Analysis 34
2.3.1. Time Domain Methods 34
2.3.2. Frequency Domain Methods 34
2.4. Frequency Domain Algorithm for Signal Comparison 35
2.4.1. Approach 36
2.4.2. Implementation 39
2.4.3. Possible Modifications 39
2.4.4. Initial Results 41
2.5. Evidence Engine Software 42
2.6. Object Model for the Evidence Engine 48
2.7. Updating Time-to-Failure Distributions 49
2.7.1. Software Description 51
2.7.2. Examples of Resulting TTFs 51
3. Consequence Engine 55
3.1. Introduction 55
3.1.1. Objectives 55
3.2. Overview 55
3.3. Approach 57
3.3.1. Schedule Module 57
3.3.2. Reliability Module 58
3.3.3. Spares Module 59
3.3.4. Cost Module 59
3.3.5. Simulation Engine 59
3.3.6. Input and Output Descriptions 61
3.4. Consequence Example 61
3.4.1. Example Input Data 61
3.4.2. Simulation Results 66
3.4.3. Example Scenario 67
3.5. Maintenance Optimization 70

3.5.1. Accessory Drive Gearbox 70
3.5.2. Sensitivity Analysis 76
3.5.3. Optimization Analysis 87
3.5.4. Analysis of Time-Change Interval 90
4. Summary 94
5. References 95
Appendix A: Software Objects in Evidence Engine 97
Appendix B: Enumerations used in the Consequence Engine 104
B.1 Equipment States 104
B.2 Event Types 105
B.3 Time Units 105
B.4 Failure Mode States 105
B.5 Preventive Maintenance Action 105
B.6 Preventive Maintenance Decision 105
Appendix C: Distributions available in the Consequence Engine 106
C.1 Fixed 106
C.2 Wear-Out 106
C.3 Exponential 107
C.4 Uniform 107
C.5 Triangular 107
C.6 Normal 107
C.7 Weibull 107
Appendix D. Schedule Module 109
D.1 Special Periods 109
D.2 Preventive Maintenance 109
D.3 Failure Modes Addressed by the PM 111
Appendix E. Reliability Module 113
E.1 Failure Modes 113
E.2 Success Paths 114
Appendix F. Spares Module 116
Appendix G. Cost Module 117
Appendix H. Input Description for the Consequence Engine 118
H.1 The Consequence Engine Uncertainty and Optimization Utility 118
H.1.1 Population File 119
H.1.2 Variable Definitions File 121
H.1.3 Baseline Input File 122
Cost Function Segment 136
Appendix I. Output Description for the Consequence Engine 137
I.1 Simulation Summary 137
I.2 Number of Failures by Failure Mode 137
I.3 Cost by Failure Mode 137
I.4 Unscheduled Downtime by Failure Mode 138
I.5 Other Output 138
Distribution 141

Table of Figures

Figure 1.1. PHM Functional Architecture 10
Figure 2.1. Bayesian Belief Network Example for ADG Fault: Metal Chips in Oil. 15
Figure 2.2. Baseline BBN, No Evidence Propagated 17
Figure 2.3. BBN with Evidence from the Magnetic Chip Detector 18
Figure 2.4. BBN Updating with Evidence from all three evidence sources 19
Figure 2.5. BBN Updating with Evidence from all three sources in "worst" states 19
Figure 2.6. BBN Updating with Prior Distribution changed 20
Figure 2.7. BBN showing influence of two failure modes on metal chips in oil 20
Figure 2.8. Simple BBN with Evidence propagated back to two Failure Nodes 22
Figure 2.9. Full BBN for Lube Problems (Metal and Nonmetal chips) in ADG Oil 23
Figure 2.10. Baseline probabilities for the Full BBN 24
Figure 2.11. Updated probabilities with two evidence sources 25
Figure 2.12. Updated probabilities with three evidence sources 25
Figure 2.13: U-matrix (left) and resulting output of clustering algorithm (right) 27
Figure 2.14: The cluster map with locations of training points for normal data (pink), and oil leak data (black) 28
Figure 2.15: Locations on the SOM of testing data for normal data (pink), and faulty oil leak data (black). 29
Figure 2.16: U-matrix (left) and output of clustering algorithm (right). Class 1 (dark blue) represents oil-leak classification. Class 2(light blue) represents tooth wear. Class 3 (green) denotes normal data files. Class 4 (orange) represents gear misalignment. Class 5 (brown) is undesignated. 31
Figure 2.17: Clustered map with locations of training data vectors: oil leak (black), normal (pink), tooth wear (red), gear misalignment (cyan). Compare with figure 2.16. 32
Figure 2.18: Clustered map with locations of testing data vectors: oil leak (black), normal (pink), tooth wear (red), gear misalignment (cyan). Compare with Figure 2.16. 33
Figure 2.19 Steps in the Frequency Domain Signal Comparison Algorithm 36
Figure 2.20. Waterfall Plot of Auto-Spectrum 37
Figure 2.21. Example Results from Signal Comparison Algorithm 41
Figure 2.22. Initialization file for Evidence Engine Demo 44
Figure 2.23. Selection of failure mode and display of parameters, PDF 44
Figure 2.24. Feature Extraction/Trend Projection 45
Figure 2.25. BBN Updating 46
Figure 2.26. Results of updated TTF distribution 47
Figure 2.27. Demonstration of Approach showing PDFs for All States 51
Figure 2.28. Normal State PDF 52
Figure 2.29. Normal State PDF Conditioned 8 Years 52
Figure 2.30. Intermediate State PDF 52
Figure 2.31. Severe State PDF 53
Figure 2.32. Combined State PDF, Probabilities = 0.333, 0.333, 0.333 53
Figure 2.33. Combined State PDF, Probabilities = 0.1, 0.5, 0.4 53
Figure 2.34. Combined State PDF, Probabilities = 0.1, 0.1, 0.8 54
Figure 2.35. Predicted Median TTF versus Normal and Intermediate State Probabilities 54

Figure 3.1. Consequence Engine Architecture 57
Figure 3.2 Simple Block Diagram Model 58
Figure 3.3. Flowchart of the Consequence Engine Simulation Logic 60
Figure 3.4. Comparison of Outage Times by System/Component 67
Figure 3.5. Daily Average Wholesale Electricity Price Forecast for 2002 68
Figure 3.6. Cost Comparison for Two Maintenance Scenarios 69
Figure 3.7. Cost of Lost Electricity Generation vs Days Delay in Maintenance 69
Figure 3.8 Accessory Drive Gearbox (ADG) for a Fixed-Wing Aircraft 71
Figure 3.9. Histogram of Annual Cost Results 79
Figure 3.10. Histogram of Annual Scheduled Downtime 80
Figure 3.11 Histogram of Annual Unscheduled Downtime 80
Figure 3.12. Partial Rank Correlations of Input Variables with Annual Cost 81
Figure 3.13. Scatter Plot of PTO Time Change Interval vs. Annual Cost 82
Figure 3.14. Partial Rank Correlations of Input Variables with Annual Scheduled Downtime 83
Figure 3.15. Scatter Plot of PTO Shaft Time Change Interval vs. Annual Scheduled Downtime 83
Figure 3.16. Partial Rank Correlations of Input Variables with Annual Scheduled 84
Figure 3.17. Scatter Plot of ADG Pump Assembly Restock Time vs. Annual Unscheduled Downtime 87
Figure 3.18 Fitness History for the Optimization Analysis 88
Figure 3.19 Annual Cost History for the Optimization Analysis 88
Figure 3.20 Annual Scheduled Downtime History for the Optimization Analysis 89
Figure 3.21 Annual Unscheduled Downtime History for the Optimization Analysis 89
Simple Block Diagram Model 114
Figure H.1 The Consequence Engine Uncertainty and Optimization Utility Main Form 118

Introduction

This report summarizes the results of a three-year LDRD project on prognostics and health management (PHM). "Prognostics" refers to the capability to predict the probability of system failure over some future time interval so that appropriate actions can be taken. An alternative definition of prognostics is the capability to predict the remaining useful life of a system. Prognostic algorithms are integrated with health monitoring (through inspections, sensors, etc.) to provide an overall PHM capability that optimizes maintenance actions and results in higher availability at a lower cost. Our goal in this research was to develop PHM tools that could be applied to a wide variety of equipment (repairable, non-repairable, manufacturing, weapons, battlefield equipment, etc.) and require minimal customization to move from one system to the next. Thus, our approach was to develop a toolkit of reusable software objects/components and architecture for their use.

We have developed two software tools: an Evidence Engine and a Consequence Engine. The Evidence Engine integrates information from a variety of sources in order to take into account all the evidence that impacts a prognosis for system health. The Consequence Engine involves algorithms to analyze the consequences of various maintenance actions. This report summarizes the capabilities we have developed, describes the approach and architecture of the two engines, and provides examples of their use.

1.1. Motivation for PHM

The motivation for Prognostics and Health Management comes from the Dept. of Energy and the Dept. of Defense under huge pressure to reduce operation and support (O&S) costs of large military or industrial systems while maintaining or increasing the availability of these systems. Many reports document the need for prognostics. DARPA's interest in developing "self-aware" systems is outlined in [Christodoulou, 2002]. Dr. Vachtsevanos at Georgia Tech has a course in diagnostics and prognostics which details many of the current approaches and algorithms [Vachtsevanos, 2002]. The Applied Research Laboratory at Pennsylvania State University has been working in the area of condition-based maintenance and machinery health monitoring for many years. They have developed many tools and approaches relating to PHM [Banks and Maynard, 2001]. The Society for Machinery Failure Prevention Technology (a division of the Vibration Institute) runs an annual meeting each year. This year, the title of the meeting was "Impact of Prognostics on Organizational Success", emphasizing the critical role of prognostics in equipment development and support [Society for MFPT, 2003].

We have based our PHM approach on existing reliability analysis tools developed at SNL over many years. Thus, our PHM approach starts with a reliability model at its core (e.g., a fault tree or block diagram model). The reliability model, a set of functionally related failure modes and associated Time-to-Failure distributions (TTF), is what we modify as health monitoring activities provide an updated picture of the health of a system. The architecture of our PHM approach is shown in Figure 1.1.

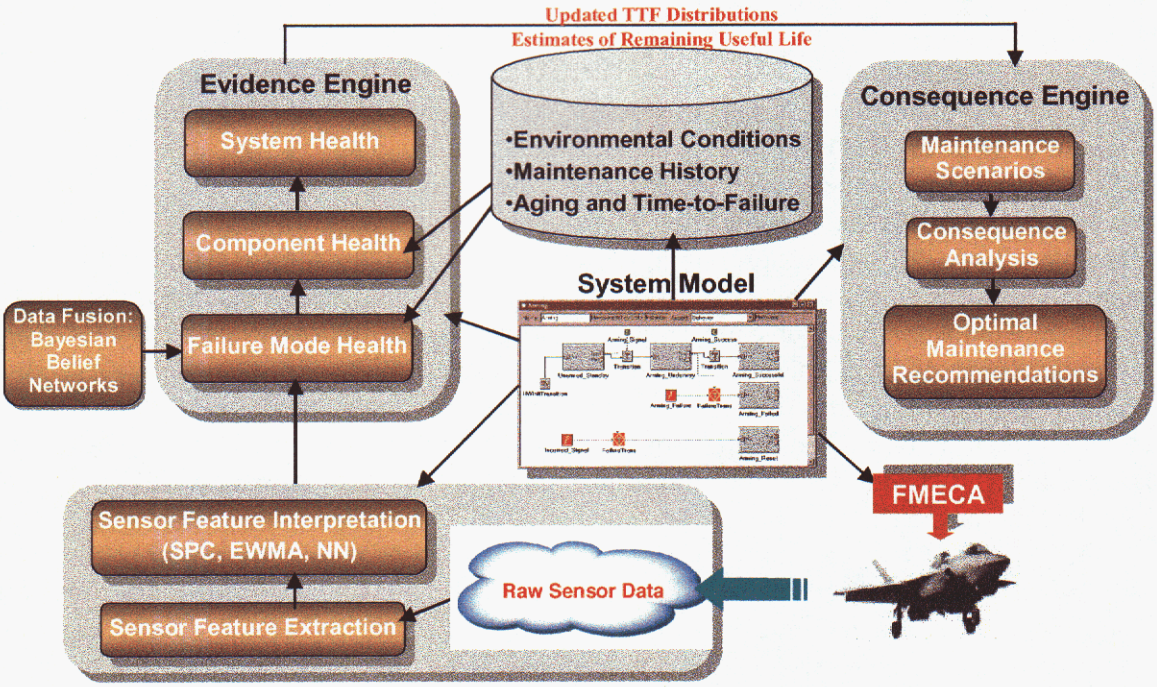


Figure 1.1. PHM Functional Architecture

1.2. PHM Functional Architecture

In our PHM architecture, a reliability model of a system is created, including a FMECA (Failure Modes Effects and Criticality Analysis). The reliability model identifies what failure modes are in the system, how they are functionally related, and useful sensors. Raw sensor data is usually high bandwidth data that must be reduced to "features" such as statistical moments, weighted moving averages, spectral density functions, etc. These features then go through an interpretation stage such as a simple threshold routine (e.g., has a feature crossed a threshold) or more complex schemes such as neural networks which classify the type of the failure mode based on some combination of sensor features. Once some "evidence" has been observed, either through sensors or through inspection/repair logs, the health of the failure modes and components is updated. This means that each failure mode and component is given a probability of being in a "good", "bad", and "intermediate" states (for example). The updating of the probability of being in various states may be done using a Bayesian belief network or another trend analysis method. Once these probabilities are updated, the time-to-failure distributions for each component are also updated. The TTF distributions give us a picture of component and system health at a particular time. These updated distributions are then sent to the Consequence engine, which runs simulations of various maintenance scenarios to determine optimal maintenance actions. The PHM architecture we show in Figure 1.1 is specific to our concept, but the steps involved (feature extraction, failure mode health updating, etc.) are fairly general and included in other architectures [Banks and Maynard, 2001; Vachtsevanos, 2002].

1.3. PHM Technology Needs

Prognostics have developed, in part, from Health Usage Monitoring Systems (HUMS), primarily used to monitor military helicopters [Hess and Hardman, 1999]. Nuclear power plant on-line monitoring has been another research area for the development of PHM systems [NERI, 2002]. The success and failure of these efforts directed some of the current research in PHM. There are several capabilities or technologies needed to perform prognostics. These include:

- Sensor feature extraction: A "feature" is a characteristic of a sensor data stream, such as hourly average, peak value, RMS, or dominant frequency. Methods to perform sensor feature extraction include time synchronous averaging, statistical analysis, peak and level-shift detection, etc. These methods currently exist.

- Sensor fusion: Sensor fusion involves combining multiple sensor features to draw conclusions. Methods to perform sensor fusion include neural networks, Bayesian belief networks, and data fusion algorithms [Roemer, Kacprzynski, and Orsagh, 2001]. Several sensor fusion methods currently exist. However, all of these methods rely heavily on having a large knowledge database to help them learn when the data may indicate a potential problem. In a new program, it may be very difficult to know enough about sensor signatures in advance to provide a rich enough database to seed the prognostics algorithms.

- Component health estimation: To estimate a component's remaining useful life (RUL) or Time-to-Failure (TTF), one needs to have models which relate the available evidence (sensor data, maintenance and/or inspection data, flight recorder parameters, etc.) to component TTF. [Engle et al., 2000; Kacprzynski et al., 2002]. Methods to do this will include trend analysis (taking sensor feature data, identifying a "bad" threshold and determining how long it will take the sensor data to reach that threshold given past history) and model-based or case-based reasoning (e.g., IF the altitude is X and the speed is Y and the sensor reading is Z and the maintenance status is W, then the component time to failure is V.) Such an expert system is likely to be huge and unwieldy.

  Component health estimation is a very difficult problem, and the science is not yet available to accurately predict TTF for many critical components.

  We do not understand how to integrate various types of evidence about a part (such as age, condition, current flight parameters, etc.) to update the effective age. Physics-of-failure models have been highly touted yet they usually are at a micro-level of analysis, not at an LRU level, and there is not yet a good bridge from the micro-level models to components. Finally, we do not understand quantitatively how load variations (in terms of running different types of missions) impacts lifetime. Aircraft parts are usually rated for a given mission profile, and a PHM system needs to be able to update TTF estimates if the aircraft sees a harsher or

easier environment than the rated standard. Such algorithms generally do not exist because they need to be based on a rich set of historical data. In this work, we have created a framework for updating state probabilities and component TTF that allows the use of fairly simple rules (e.g., if the operating condition is in state X, increase the failure rate by a percentage Y). The structure can allow more complex algorithms as we obtain data to develop and justify it.

- System health: Once component lifetimes are determined, it is fairly straightforward to input these to a system reliability model to determine overall system availability, downtime, and projected costs.

- Determination of consequences and optimal strategies. Once component and system health are determined, the capability to simulate potential scenarios and choose the optimal strategy is needed (e.g., what happens if the part is not replaced at all, if it is replaced immediately, or at the next scheduled maintenance.) The Consequence Engine analyzes possible scenarios accounting for projected TTF, parts availability, planned equipment use schedule, and mission profile. We are the only group that we know of developing a simulation-based approach for consequence analysis. The only other "consequence analysis" we are aware of is a cost-benefit PHM model developed by Impact Technologies [Kacprzynski, Roemer, and Hess, 2002].

1.4. LDRD Focus

We have learned much about the state of prognostics and various applications. Over the course of this LDRD, we have changed emphasis somewhat and have put more of our efforts in developing the Consequence Engine and less on developing the Evidence Engine, for two reasons:

1) The Consequence Engine utilizes Sandia's strengths in reliability analysis and simulation. Also, the Consequence Engine is a general tool applicable to a wide range of PHM applications.
2) There are limits to the extent that general-purpose tools can be designed for feature extraction and evidence integration since these functions tend to be very system-specific. For this reason, additional algorithms (or a customization of what we already have) will need to be developed to update failure mode state probabilities and TTF distributions for specific applications.

During the first year of this LDRD, we designed object models for each of the Engines, and developed algorithms and approaches. In the second year, we continued implementation and added functionality. During this third year, we have added optimization and a spares modeling capability to the Consequence Engine, and we added some additional signal processing and feature classification capabilities for Evidence integration.

This report is divided into two large sections: The Evidence Engine and the Consequence Engine. Since the first year report (Final Report, LDRD 01-0329) documents the object models and computer code behind both software engines extensively, we have chosen not

to duplicate all of that information here. The second year report (Final Report, LDRD02-1277) outlines the Bayesian Belief Network approach in detail, presents the "success path" modifications to cut sets that allow calculation of various operational states in the Consequence Engine, and presents a peak finding algorithm for vibration analysis. This report refers to the previous two reports when appropriate, but we have tried to replicate some of the material above so that this can be a stand-alone final report for this LDRD project.

Evidence Engine

The Evidence Engine integrates information from a variety of sources in order to take into account all the evidence that impacts a prognosis for system health. We have developed a prototype Evidence Engine. We have also developed some modular software objects that perform a "step" in the evidence integration stage (such as taking the value of a sensor feature and translating it to failure mode state probabilities). Finally, we have developed some examples to demonstrate the usefulness of various techniques. The outline of this section of the report is as follows: first, we discuss Bayesian Belief networks and feature classification using Kohonen self-organizing maps (SOMS). These are two methods useful to evidence integration. We also discuss some work in the area of vibration analysis, since some of the current application interest is in the area of rotating equipment. Then, we present an overview of the evidence engine software and some of the modular objects developed as part of this effort. Finally, we discuss our approach for updating Time-to-Failure distributions.

1.5. Bayesian Belief Networks

1.5.1. Bayesian Belief Networks - Background

A Bayesian Belief Network (BBN) is a graphical network that represents probabilistic relationships among variables. BBNs enable reasoning under uncertainty. With BBNs, it is possible to articulate expert beliefs about the dependencies between different variables and to propagate consistently the impact of evidence on the probabilities of uncertain outcomes, such as 'future system reliability.'

A BBN is a special type of diagram (called a graph) together with an associated set of probability tables. The graph is made up of nodes and arcs where the nodes represent uncertain variables and the arcs the causal/relevance relationships between the variables. The main use of BBNs is in situations that require statistical inference: in addition to statements about the probabilities of events, the user knows some evidence, that is, some events that have actually been observed, and wishes to infer the probabilities of other events, which have not as yet been observed. A BBN uses conditional probability tables to calculate the probabilities of various possible causes being the actual cause of an event. [Jensen, 2001]

A major benefit of Bayesian inference over 'classical statistical inference' (which deals with confidence levels rather than statements of probability) is that it explicitly describes the fact that observation alone cannot predict the probability of unobserved events. In the Bayesian interpretation, a probability describes the strength of the belief which an observer can justifiably hold that a certain statement of fact is true (subjective probability). The subject, after observing the outcome of an 'experiment' (i.e., collecting new data), updates the belief held before the experiment (the 'prior probability'), producing a 'posterior probability'. The need to assume prior beliefs is a key part of Bayesian inference. The conditional probability tables must be filled in with reasonable

estimates, and it is not always easy to obtain sensible prior probabilities, even from domain experts.

Until recently, the computation necessary to calculate the posterior probabilities for a BBN were quite difficult to implement, even for small problems. The general problem of performing such computations is known to be intractable (formally, it is known to be an NP-hard problem). The 1990s has seen the development of many tools which incorporate fairly efficient solution algorithms for BBNs. We are using one such tool called HUGIN, for "Handling Uncertainty in Generalized Inference Networks."

1.5.2. Bayesian Belief Network – Example

We developed a Bayesian Belief network of an Accessory Drive Gearbox (ADG) on a military aircraft. This is an important system for prognostics, since it is a flight critical system. Maintenance currently requires ADG replacement at a certain number of flight hours. Condition of the gearbox at overhaul indicates that there is much useful life remaining, but to extend the time change, a system must be in place to monitor operational life and mitigate premature catastrophic failures.

We are starting to get data about wear mechanisms and failure modes of the ADG. The best example to focus on for the ADG is oil contamination, since the current inspection practices examine three things which might indicate metal chips in the oil: the magnetic chip detector, the "delta-p" indicator on the oil filter (which is a differential pressure sensor across the filter), and visual inspection of the oil filter.

A simple BBN is shown in Figure 2.1. Note that the node labels have prefixes: S_ indicates a sensor node (sensor nodes may also involve tests), C_ indicates a condition, and F_ indicates a fault. The top node in this BBN is the fault condition of metal chips in the oil. This BBN does not indicate what type of failure modes may have caused the metal chips to be present – this will be presented later.

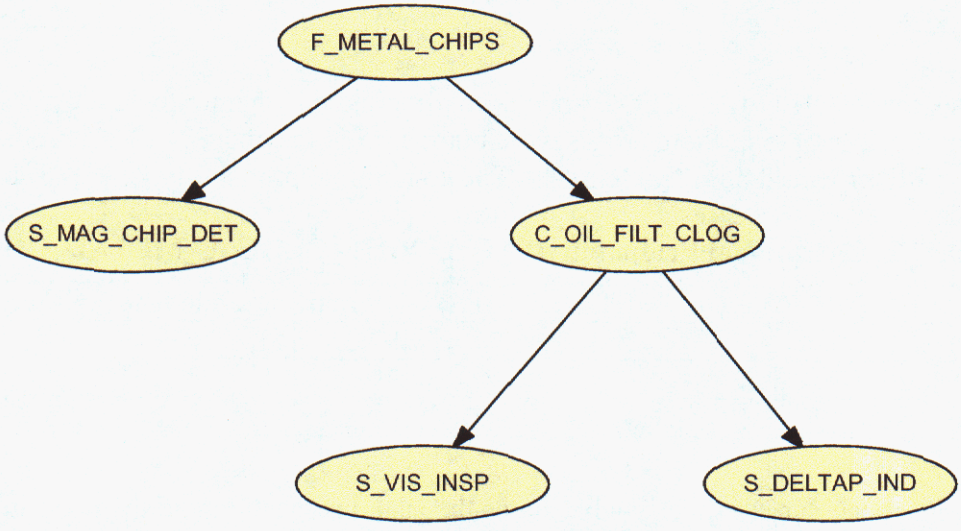


Figure 2.1. Bayesian Belief Network Example for ADG Fault: Metal Chips in Oil.

The following tables present the data that was used to populate the BBN. Note that this data may not be representative of an actual gearbox: the idea was to test the BBN and make sure that the updated distributions were consistent with the assumptions. This example will give the user a feel for the amount of conditional probability estimates necessary for this type of analysis.

First, the prior probabilities of having metal chips in the oil are needed. The F_METAL_CHIPS node has three states, with the following initial probabilities:

Prob. (Normal) = 0.9
Prob. (Intermediate) = 0.08
Prob. (Severe) = 0.02

Thus, in absence of any information, we assume that there is a 90% chance that the oil will have a "normal" amount of chips, an 8% chance it will show an intermediate amount of chips, and a 2% chance that there will be enough chips indicating a severe amount of wear.

The conditional probability table for the magnetic chip detector is given in Table 2.1. This reads as follows: The magnetic chip node will be in the "clean" state with a 95% probability, given that the underlying metal chips fault state (F_METAL_CHIPS) is normal. The magnetic chip node will have a few chips with a 4% chance if the underlying metal chips state is normal, and will have many chips with a 1% chance given the underlying metal chips state is normal.

Table 2.1. Conditional Probability(S_MAG_CHIP_DET|F_METAL_CHIPS)

Prob(S_MAG_CHIP_DET|F_METAL_CHIPS)	Normal	Intermediate	Severe
Clean	.95	.05	.05
Few Chips	.04	.8	.1
Many Chips	.01	.15	.85

Likewise, the C_OIL_FILT_CLOG node is conditioned on whether there are metal chips in the oil. For example, if there are a severe number of chips, there is a 95% probability that the oil filter would be fully clogged. The conditional probability table is Table 2.2:

Table 2.2 Conditional Probability(C_OIL_FILT_CLOG |F_METAL_CHIPS)

Prob(C_OIL_FILT_CLOG|F_METAL_CHIPS)	Normal	Intermediate	Severe
Clean	.9	.1	
Partial Clog	.1	.8	.05
Full Clog		.1	.95

Finally, it is necessary to specify what states the visual inspection and the pressure drop (delta-p) indicator would be in, conditional on the states of the oil filter. These are given in Tables 2.3 and 2.4. For example, the probability that one would have a popped delta-

p indicator (excessive pressure drop across the oil filter) if the filter were partially clogged is 60%.

Table 2.3. Conditional Probability: Prob(S_VIS_INSP |C_OIL_FILT_CLOG)

Prob(S_VIS_INSP|C_OIL_FILT)	Clean	Partial Clog	Full Clog
No Problem	.95	.3	.05
Problem	.05	.7	.95

Table 2.4. Conditional Probability: Prob(S_DELTAP_IND |C_OIL_FILT_CLOG)

Prob(S_DELTAP_IND|C_OIL_FILT)	Clean	Partial Clog	Full Clog
No Pop	.98	.4	.02
Pop	.02	.6	.98

The following examples show how Bayesian networks may be used to "propagate evidence" from the "leaf nodes", in this case the sensor nodes, to the "root node", in this case the fault condition of metal chips. Figure 2.2 shows the "baseline" BBN with no additional evidence gathered. In this case, the probability of the F_METAL_CHIPS node being in a normal, intermediate, or severe state is 90%, 8%, or 2% respectively (these were the "prior" or initial probabilities). These probabilities are then combined with the conditional tables above to determine the probabilities of being in various states for the other nodes. For example, the magnetic chip detector has an 86% chance of being in a normal state, a 10% chance of having a few chips, and a 4% chance of having many chips.

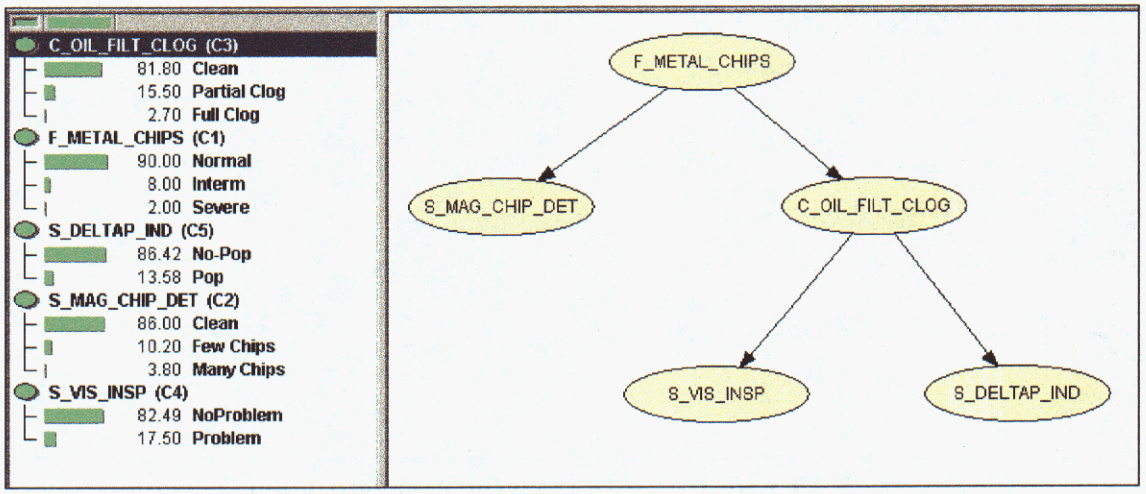


Figure 2.2. Baseline BBN, No Evidence Propagated

If some evidence is gathered, this can be used to "update" the BBN to calculate the posterior probabilities on the state for F_METAL_CHIPS. Look at the example in Figure 2.3. If we see many chips in the chip detector (so the probability of being in the state "many chips" is 100% for the S_MAG_CHIP_DET node, as indicated by the red line on the left hand side of the picture), the probability of metal chips being in a severe state goes up from the baseline 2% to nearly 45%. To understand how this is calculated, we use Bayes rule: Prob(A|B) = P (AB)/P(B).

The probability of the magnetic chip detector node being in the "many chip" state is:

Prob(S_MAG_ CHIP_DET = "Many chips") =
Σ (Prob S_MAG_CHIP_DET = "Many chips| F_METAL_CHIPS) *
Prob(F_METAL_CHIPS = "normal, intermediate, or severe")
= 0.01*.9 + 0.15*.08 + .85*.02 = .038

where the sum is over all states of the F_METAL_CHIPS node. Then, the conditional probabilities are reversed by marginalizing the probability (F_METAL_CHIPS = severe) from the joint probability: P(AB)/P(A) = P (B/A). In this case, the joint probability is:

(S_MAG_CHIP_DET=Many chips AND F_METAL CHIPS = severe)=
0.85*.02=.017.

We then obtain the conditional probability:

Prob(F_METAL_CHIPS = severe|S_MAG_CHIP_DET = Many chips) =
.017/.038 = .447 ≈ 45%.

This example shows how "evidence", in this case the sighting of many chips on the magnetic chip detector, can be propagated in a Bayesian belief network to update one's "prior" beliefs about the underlying state of nature. In this example, if we see many chips in the chip detector, our "posterior" belief in the state of metal chips in the lubricant goes from a prior of 2% to a posterior estimate of 45%.

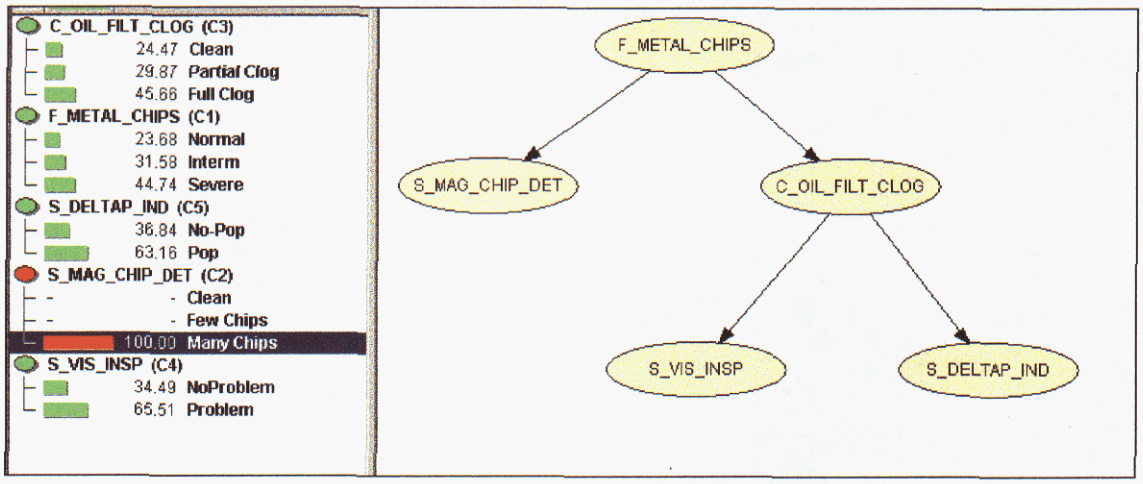


Figure 2.3. BBN with Evidence from the Magnetic Chip Detector

To extend the BBN updating example, see Figure 2.4. Here we have gathered information from all three pieces of evidence: the delta-p indicator is popped, there are a few chips on the chip detector, and visual inspection indicates that there is a problem. In this case, the updating procedure is more complex, because there are multi-variate joint distributions. The BBN has to calculate many sets of distributions involving P(A|B,C) = P(A,B|C)|P(B|C), etc. In the example shown in Figure 4, the updated belief about the state of metal chips in the lubricant shows an 89% probability of being in an intermediate

state, a 6% probability of being in a severe state, and a 5 % probability of being in a normal state. Note that the large posterior probability for the intermediate state is influenced by the high probability of the magnetic chip detector being in the few chips state given the intermediate state of the metal chips (Table 2.1).

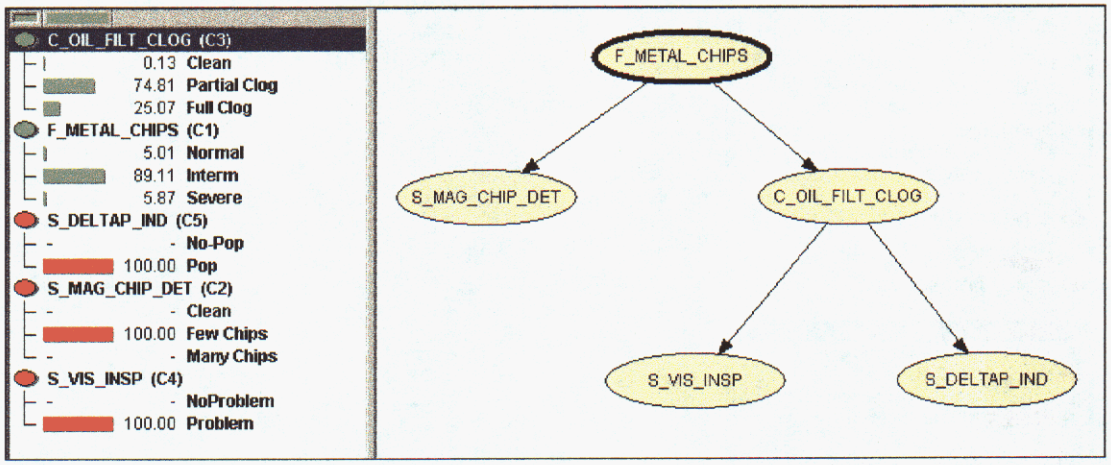


Figure 2.4. BBN Updating with Evidence from all three evidence sources

Figure 2.5 shows the same situation as Figure 2.4, only the evidence from the chip detector now shows many chips instead of few chips. This raises the posterior probability of the lubricant being in a severe state from 6% to nearly 73%. The importance of this example is to demonstrate how important the conditional probabilities are in determining the posterior distribution during updating.

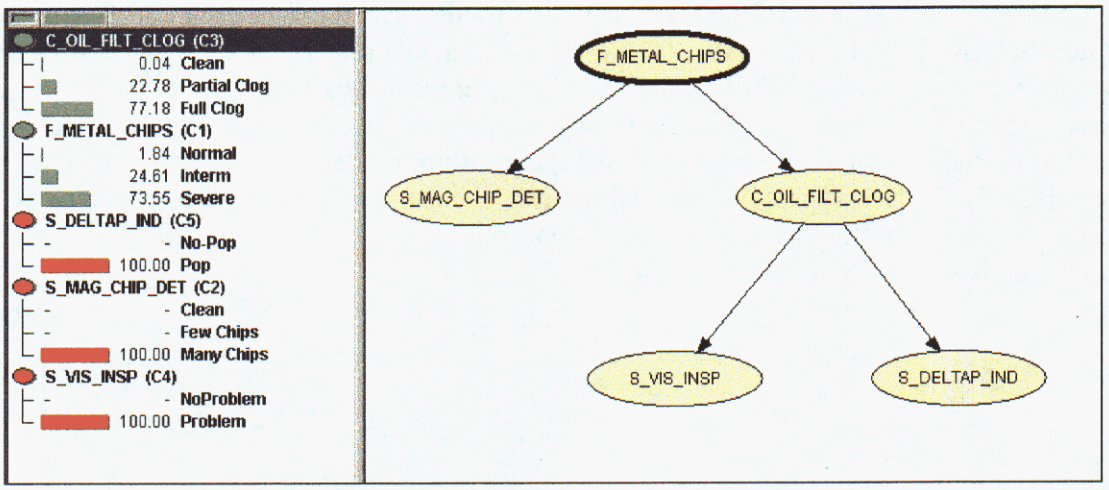


Figure 2.5. BBN Updating with Evidence from all three sources in "worst" states

Finally, the example in Figure 2.6 shows that the prior probabilities also play an important role in the calculation of the posteriors. Figure 2.6 shows the same example as Figure 5, except that the priors on the "F_METAL_CHIP" state have been changed. The prior probability of normal is still 90%, but the probability of being in an intermediate state of degradation was decreased from 8% to 5%, and the probability of severe number

of chips in the oil has increased from 2% to 5%. The result of this is that the updating yields a much higher posterior probability of being in a severe state: it has risen from 6% to 18%, while the probability of being in an intermediate state has decreased from 89% to 74%.

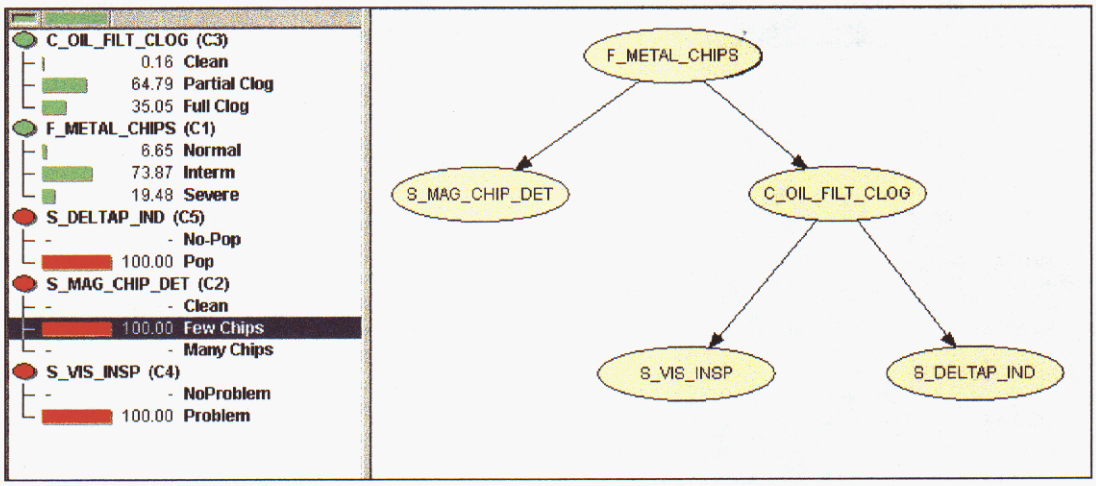


Figure 2.6. BBN Updating with Prior Distribution changed

The examples presented in Figures 2.1-2.6 show how the BBN is updated, and the importance of both the prior distributions and the conditional probabilities in the updating process.

After doing this initial analysis, we decided to make the Bayesian network more realistic and informative. The purpose of using a BBN in a prognostic or diagnostic application is to update the probabilities of a particular failure mode occurring. In the example for the state of lubrication of the accessory drive gearbox, metal chips in the oil may indicate a wide variety of potential failure modes, such as gear wear, bearing wear, metal contaminants remaining from the manufacturing process, or seal wear. A priori, we do not have much information about which of these failure modes would be more likely. We created a BBN to represent these failure modes influencing the state of particles in the oil. A simple BBN just to explore the concept of two failure modes influencing one fault state is shown in Figure 2.7:

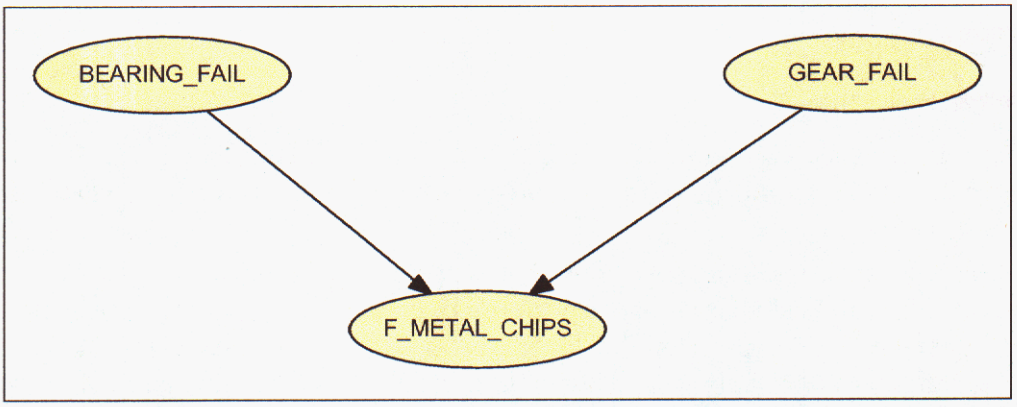


Figure 2.7. BBN showing influence of two failure modes on metal chips in oil

Here the baseline probabilities for both bearing failure and gear failure are given as 0.01 failure, 0.99 no failure. The conditional probability table for F_METAL_CHIPS is given in Table 2.5.

Table 2.5.Conditional Probability(F_METAL_CHIPS|BEARING and GEAR FAIL)

GEAR_FAIL	YES		NO	
BEARING_FAIL	YES	NO	YES	NO
Normal	0	0	0	1
Intermediate	0	.2	.4	0
Severe	1	.8	.6	0

Note that these probabilities are not symmetric: if there is gear failure but not bearing failure, the probability of a severe state of metal chips in the oil is 80%, while if there is bearing failure but not gear failure, the probability drops to 60%. The question is: If someone "observes" severe metal chip conditions in the oil, what are the probabilities that the gears or bearings have failed? To determine this, we use the compound Bayes rule: P(A|BC) = P(ABC)/P(BC). We take the conditional probabilities in Table 2.5 and multiply them by P(BC) to get the joint distribution P(ABC). Then, we divide the joint probability table by P(A) to obtain P(BC|A), then finally sum over the individual distributions to get P(B|A) and P(C|A). First we need to construct the joint probability table shown in Table 2.6. Note that:

P(BC) = Prob (GearFail = BearFail = YES, GearFail=YES and BearFail = NO, GearFail = NO and BearFail = YES, GearFail = BearFail = NO)
== (.0001, .0099, .0099, .9801).

Table 2.6. Joint Probability (F_METAL_CHIPS and BEARING and GEAR FAIL)

GEAR_FAIL	YES		NO		TOTAL
BEARING_FAIL	YES	NO	YES	NO	P(METAL_CHIPS)
Normal	0	0	0	.9801	.9801
Intermediate	0	.00198	.00396	0	.00594
Severe	.0001	.00792	.00594	0	.01396

Thus, to perform the conditioning to determine P(BC|A), we take Table 2.5 and divide the joint probabilities by the marginal probability of A given in the last column. We get Table 2.7.

Table 2.7. Conditional Probability (BEARING and GEAR FAIL|F_METAL_CHIPS)

State of METAL_CHIPS	Normal	Intermediate	Severe
GearFail = YES, BearFail = YES	0	0	.0072
GearFail = YES, BearFail = NO	0	.333	.5673
GearFail = NO, BearFail = YES	0	.667	.4255
GearFail = NO, BearFail = NO	1	0	0

Thus, if the condition observed is a severe state of metal chips, the probability that there is gear failure is the sum of the first two rows of the last column, where gear failure = yes, or .5745. This is shown in the BBN depicted in Figure 2.8.

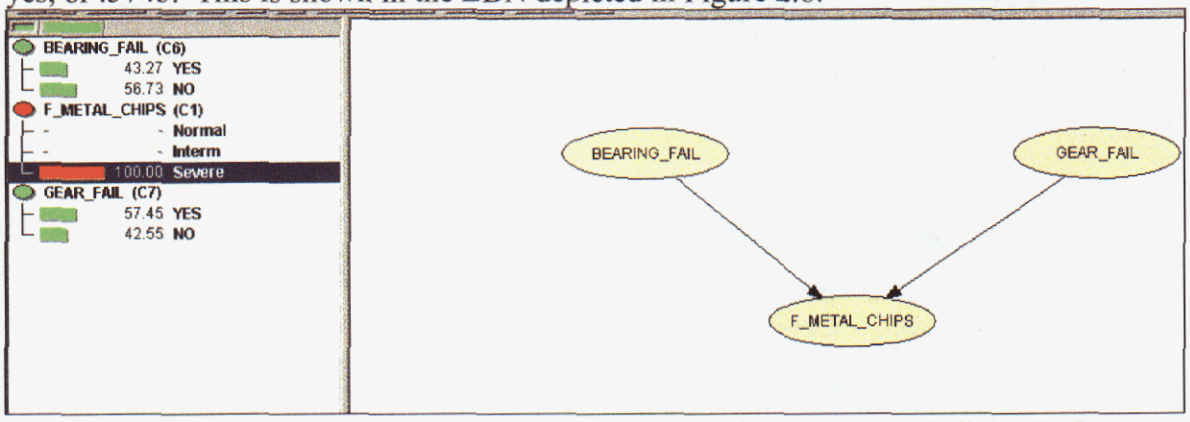


Figure 2.8. Simple BBN with Evidence propagated back to two Failure Nodes

Notice in this example, because the conditioning in Table 2.5 is not symmetric, the posterior probabilities of each failure mode occurring are different: in this example where F_METAL_CHIPS is in a severe state, gear failure will occur with probability 57% but bearing failure will occur only with a probability of 44%. If we had made the conditional probabilities symmetric, then the posterior probability of each failure mode occurring would be equal.

Finally, we made a detailed and realistic example incorporating both the ideas of evidence in Figures 2.1-2.6 and the idea of determining likely causes in Figures 2.7-2.8. To make the example more realistic, we partitioned the chip types into ferromagnetic and non-ferromagnetic chips. The new BBN is shown in Figure 2.9.

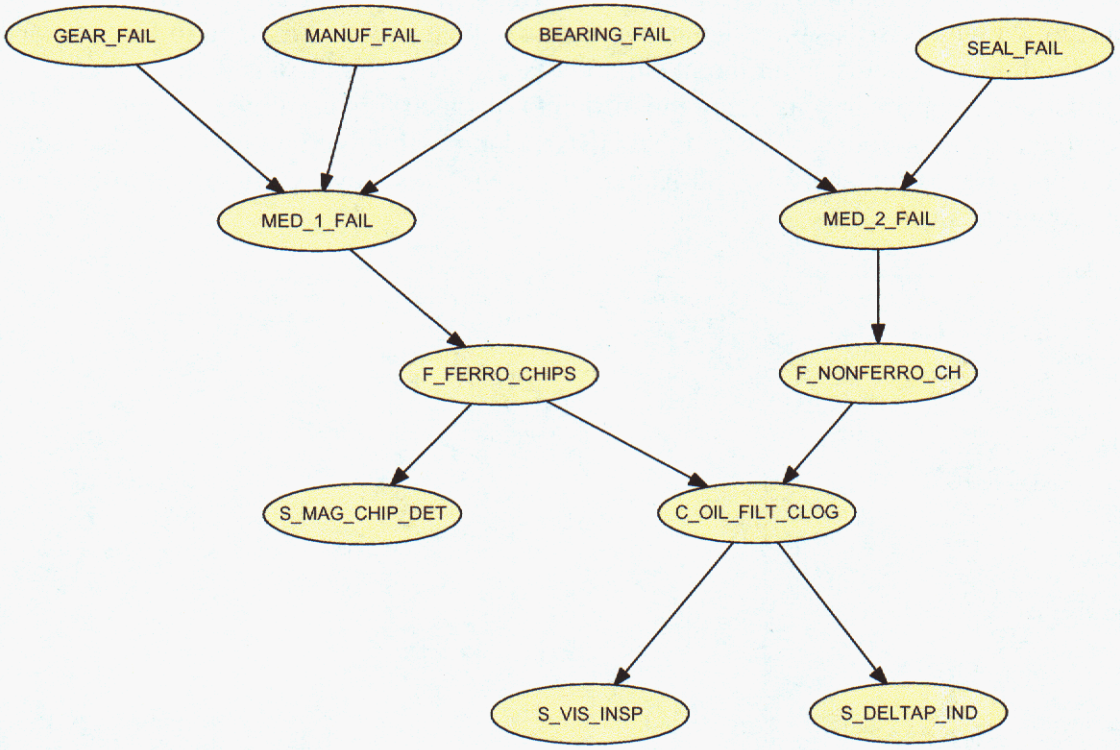


Figure 2.9. Full BBN for Lube Problems (Metal and Nonmetal chips) in ADG Oil

Figure 2.9 shows four failure modes influencing the presence of ferromagnetic or nonferromagnetic chips (carbon) in the oil: gear failure, bearing failure, seal failure, and manufacturing process failure. Bearing and seal failure influence the nonferromagnetic chip level, since there are carbon bearings and seals in the ADG. Bearing, gear, and manufacturing failures influence the ferromagnetic chip level (it is our understanding that there are both metal and carbon bearings). The gates "MED_1_FAIL" and "MED_2_FAIL" are mediating gates that act as "OR" gates: if one of the failure modes has occurred, the FAIL=TRUE happens with probability of 1, otherwise FAIL=FALSE occurs with probability of 1. The baseline probabilities of the four failure modes are all identical, with a failure probability of 0.01. The probabilities of the evidence sources propagating up to F_FERRO_CHIPS are basically the same as those presented in Tables 2.1-2.4, except that the probability of the oil filter being in a clogged condition is now dependent on both the ferromagnetic chip node and the non-ferromagnetic chip node.

Figures 2.10-2.12 show the types of analysis that can be done on this full BBN. If there is no evidence propagated, the baseline normal probabilities of having ferromagnetic or nonferromagnetic chips are both near 88%. If visual inspection shows a problem and the delta-p indicator pops indicating an oil filter clog, the failure probabilities on all the failure modes increase (Figure 2.11). For example, the seal failure mode probability increases to 5.5%. If, in addition, the magnetic chip detector has many chips, the failure probabilities on the three failure modes feeding into ferromagnetic chips increases. For example, gear failure increases to 25%. However, the seal failure probability decreases, in this case to 1.4%, because it is less likely to be seal failure if the magnetic chip

detector indicates many chips (Figure 2.12). This is the type of analysis that can be performed with a BBN. Bayesian networks are quite useful as a method to fuse sensor data and update beliefs about the likelihood of a failure cause. BBNs do have their limitations, the primary one being the difficulty of specifying accurate prior and conditional distributions. We feel that BBNs, along with other methods such as neural networks, regression analysis, and AI expert system tools, have an important role to play in prognostics.

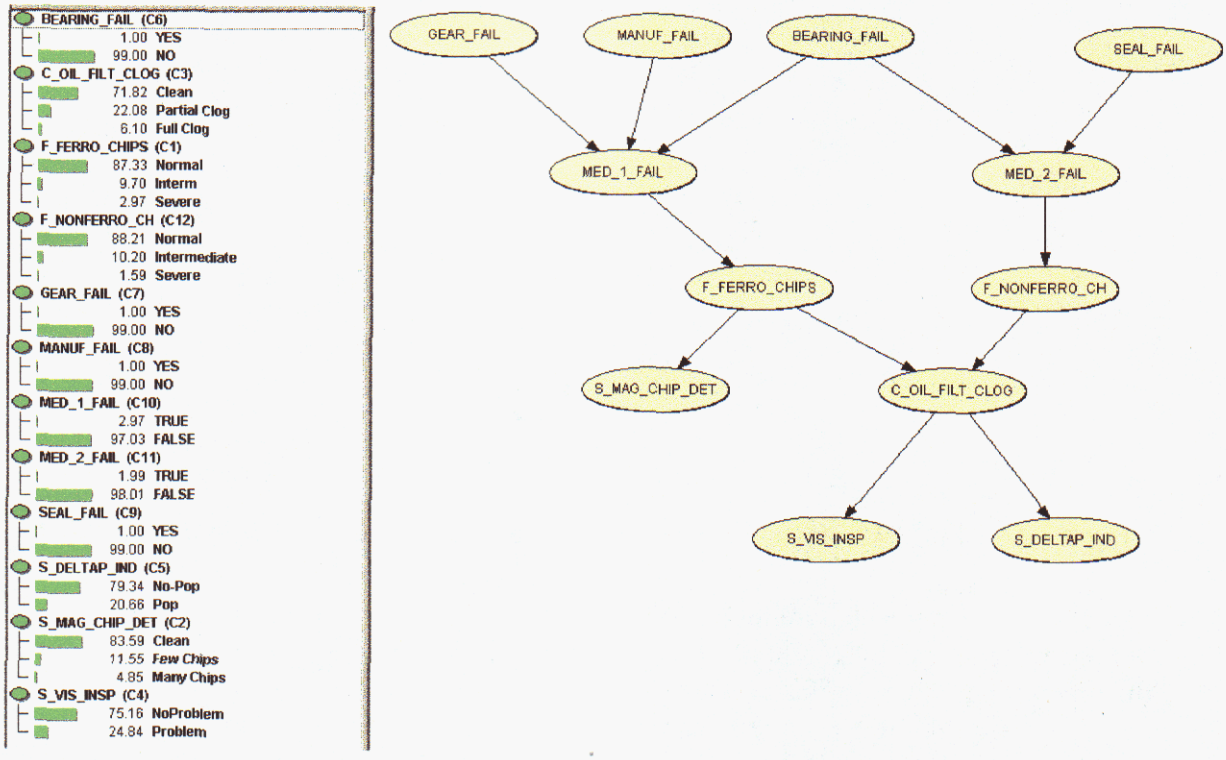


Figure 2.10. Baseline probabilities for the Full BBN.

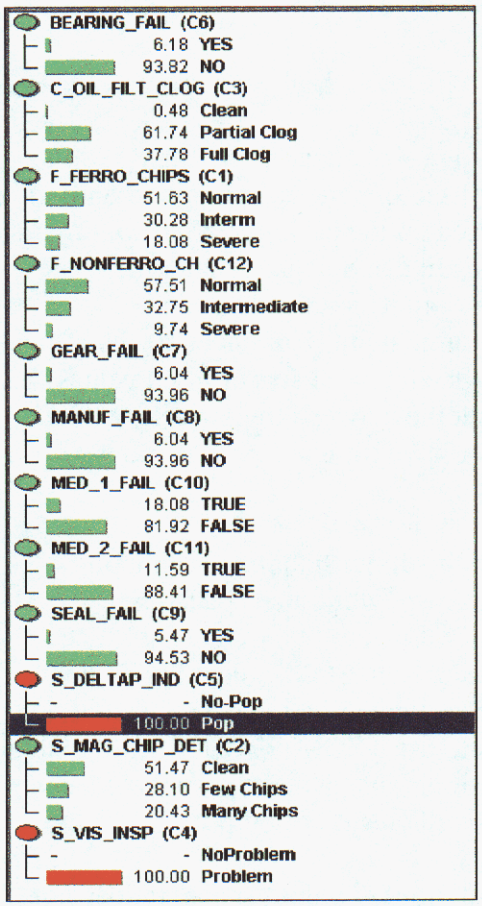


Figure 2.11. Updated probabilities with two evidence sources

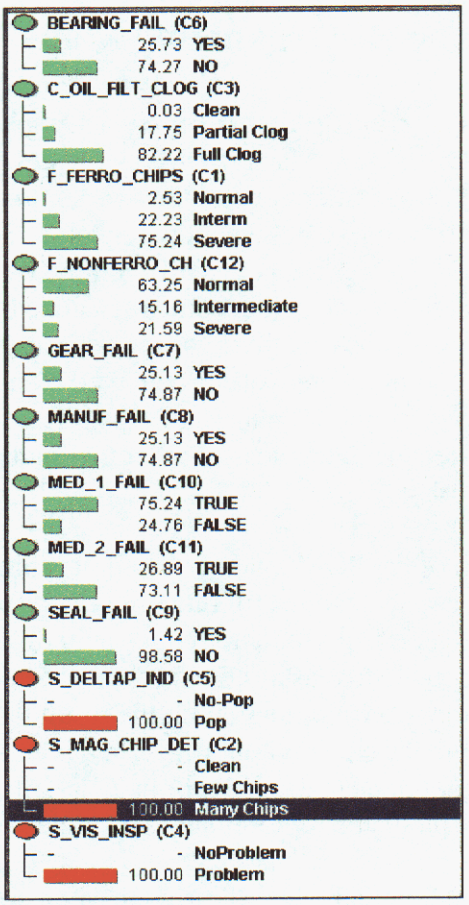


Figure 2.12. Updated probabilities with three evidence sources

1.6. Self-Organizing Maps

1.6.1. Background on Self Organizing Maps

Self Organizing Maps (SOMs) were developed by Teuvo Kohonen, in the early 1980's. SOMs are similar to neural networks, in that they are used in pattern recognition and classification. Unlike neural networks, SOMs are based on the principle of *competitive learning*. Over the course of training, by means of positive and negative lateral interactions, one cell becomes sensitized to a region of input signal values, and suppresses the sensitivity of the cells around it to the same input [Kohonen, 2001]. Thus, each cell in the network is activated by a different constellation of sensor input values. The *Self* in Self Organizing Maps refers to the fact that the network trains itself, without any preconceived ideas of what the final outcome should be (unlike many neural networks).

Much of the SOM's power lies in its ability to reduce the dimensionality of an input vector space, while still retaining the distance relationships within that space. For instance, it is possible to reduce a 50 dimensional space down to two dimensions, and still be able to classify the data using a SOM. For this reason, SOMs generally do not provide a 1-1 mapping, with the result that more than one combination of inputs will activate the same output cell. However, if one wants to use a SOM for classification, this does not present a problem. In a classification application, one generally wants to have many inputs be represented as a single, or at most a small cluster, of outputs.

A SOM, in and of itself, provides a mapping from and input to output space. It does not classify data. However, any standard clustering algorithm can be implemented in a post-processing phase to achieve data classification by placing labels on cells. The computational costs of a SOM and clustering algorithm are incurred in the training phase, with very little computation required to classify new data as it becomes available. Thus, SOMs are suitable for near real time applications.

In the following sections, an analysis of gearbox data is undertaken using SOMs. In the first section, classification is performed using actual data: healthy system data, and data representing one failure mode. In the next section, available documentation on diagnostics is utilized on healthy system data to simulate additional failure modes. A different SOM is trained in the classification of the simulated data.

1.6.2. SOM Analysis of Available Data

The first SOM generated in this project was used to examine actual gearbox data. Six data files were available which contained instances when the system was functioning normally. In addition, nine data files were used which demonstrated the operating conditions of the system during an oil leak. In all cases, data used represented 4096 samples, or 85.3ms.

The development portion of the SOM/clustering analysis was performed using the software package MATLAB. A free version of a MATLAB SOM toolbox, developed by the students of the SOM inventor, was available to download without licensing

restrictions at Helsinki University of Technology www.cis.hut.fi/projects/somtoolbox. The Helsinki SOM toolbox was found to be adequate for the scope of this project. An additional clustering algorithm, based on the k-means clustering technique, was created on-site to work with the SOM toolbox.

Due to the small amount of actual data that was available, two data files each of the normal data and oil leak data were used to train the SOM. The remaining data was retained for testing the trained network. Each data file was split into two pieces, and 49 time samples from each half were extracted. This created a total of 8 vectors of 49 samples each. A final element in each vector was the mean value of the magnitudes of the FFTs of the original files. Thus, from 4 data files, a total of 8 training vectors were created, each 50 points in length, with 2 training vectors being constructed from each data file.

Based on the eigenvalues of the available data, the SOM map size was selected to be 8x2. The SOM was allowed to train without knowledge of which data vectors represented good and bad data. In this phase of development, a basic k-means clustering algorithm was applied to the resulting map, with the request that there be only two resulting classes. The results are shown in the figure below. On the left, one resulting output of the SOM, the uniform distance matrix, or U-matrix, is displayed. Blue areas represent where the data is close together in feature space, red colors represent data that is far apart. Note how the dimensionality of the U-matrix is greater than 8x2. Since the U-matrix represents distances between cells, its dimension is (2*8-1)x(2*2-1).

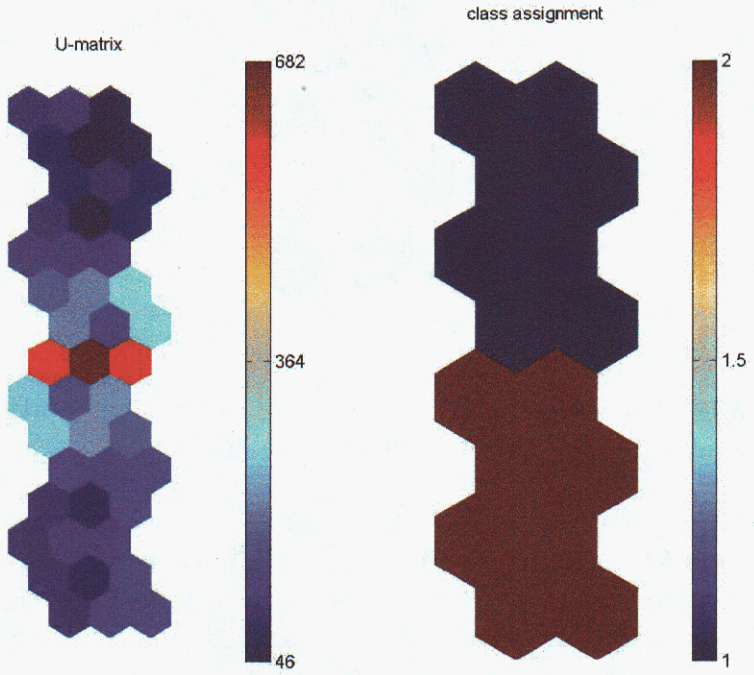


Figure 2.13: U-matrix (left) and resulting output of clustering algorithm (right).

The results of the clustering algorithm are displayed on the right. There are two classes: 1(blue), and 2(brown). If the algorithm is successful, these should represent the two conditions of the input data.

Figure 2.14 shows the locations of the training data on the clustered map. The mapped locations of the files are superimposed upon the background. The pink cells represent the locations of normal data vectors, black cells represent the locations of oil leak data vectors. A comparison with the figure above shows the data is correctly separated into two classes. Note how there were four training vectors for each state of the data, but there are only three colored cells representing each. Recall from the previous section that a SOM, due to its ability to compress the dimensionality of a feature space, does not necessarily create a 1-1 mapping. In this case, on two separate occasions, there are two data vectors which are mapped to the identical cell.

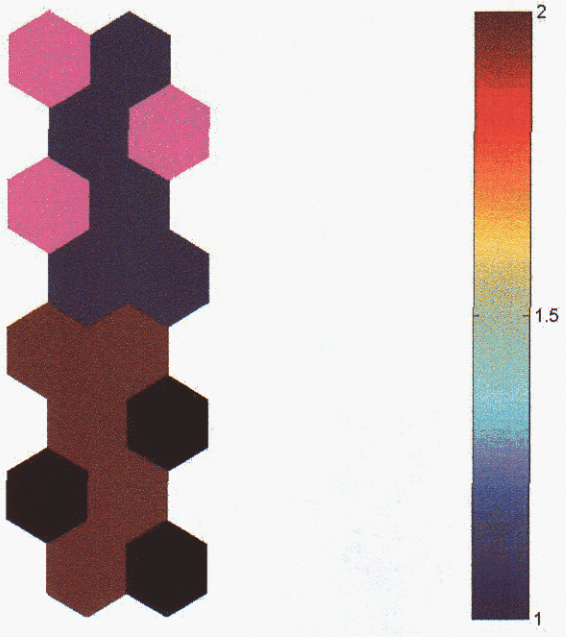


Figure 2.14: The cluster map with locations of training points for normal data (pink), and oil leak data (black).

Ultimately, the test is to see how well the SOM can do on data for which it was not trained. Four normal and seven of the available faulty data files were retained for this purpose. The testing data were processed in the same way as the training data and their locations are displayed on the figure below. Again, the data files are correctly plotted to two separate classes.

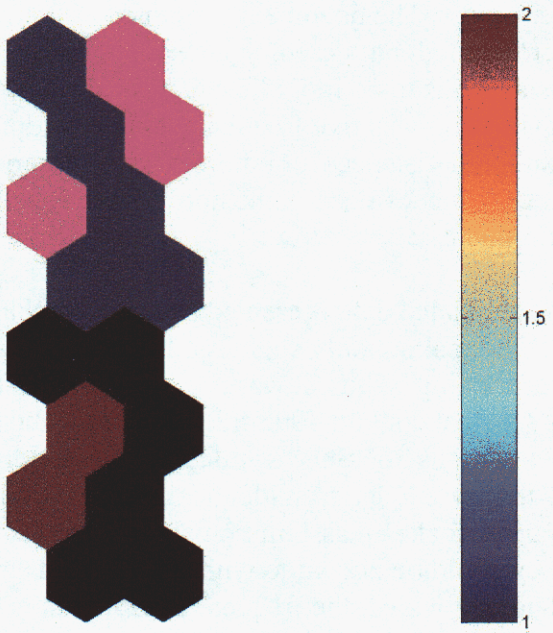


Figure 2.15: Locations on the SOM of testing data for normal data (pink), and faulty oil leak data (black).

The SOM/clustering algorithm combination had little trouble distinguishing between normal and faulty oil leak data files when actual data was employed. The classification of the testing data requires processing it through a matrix. It is nearly instantaneous when graphical display methods are not utilized. Thus, the algorithm could be programmed to produce a single digit output corresponding to a particular class and the algorithm could be used in near real time applications.

1.6.3. SOM Analysis of Simulated Faulty Data

It was desired to apply the SOM technique in the diagnoses of other types of faults. Unfortunately, in the case of the gearbox data we had, no other data showing actual examples of faults was accessible. However, a diagnostic chart demonstrating examples of faulty data signatures was available [Berry, 2003]. Based on the best available knowledge of what various faults should look like, the available normal data files were manipulated to resemble faulty data files in the following way.

Gear misalignment: This fault is manifested by the excitation of harmonic frequencies of the gear mesh frequency. No data was given in [Berry, 2003] concerning to what extent the harmonics are excited. Based on an examination of the frequency domain graph given in the reference, the gear misalignment simulation data were created by taking an FFT of the normal data files and boosting the 1^{st}-3^{rd} harmonics by a factor of 1.5. No other modifications were made.

Tooth wear: This fault is manifested by the excitation of sidebands on either side of the gear mesh frequency. The document contained no information on how much the sidebands were excited, but based on an examination of the plot provided, it was decided to excite the sidebands to no more than 60% of the amplitude of the gear mesh frequency. Each created sideband was five frequency bins in width and tapered. The sidebands were placed the minimum distance from the gear mesh frequency to allow them to be separable at the sampling frequency used in the normal data files. No further modifications were made to the normal data files.

The use of the simulated data presented a particular challenge for the SOM. A faulty data set was created by making only slight modifications to a normal data set, and both would be required for training. Thus, it was necessary for the SOM to make distinctions between data sets that contain 4096 frequency bins and may differ in as little as three. Under these conditions, the SOM would likely map normal and faulty data to the same cell, and thus they would have an identical classification. In order to get past this hurtle, it was necessary to make an assumption about the data. It was assumed that the gear mesh frequency could be known to within 10Hz of its true value. If an FFT is performed on the actual data provided, the frequency resolution is 11.7 Hz. Since 10Hz is less then 11.7 Hz, we could assume the gear mesh was known to within one frequency sample. The assumption allowed for the generation of extra data points created by calculating the ratio of values between the gear mesh frequency and its harmonics. Such a calculation could be a great asset in detecting gear misalignment.

The data used in the SOM training phase consisted of eight files: the four files from the previous SOM, two files created by corrupting two normal files to simulate gear misalignment, and two files created by corrupting two normal files to simulate tooth wear. Data input vectors to the SOM were created in the following way. First, an FFT was performed on all the input files. To reduce the size of the vectors, the frequency content was averaged over a length of 40 bins and only magnitudes were included. Next, three parameters relating to the frequency ratios of the harmonics to the fundamental attached. Finally, a magnification calculation on the gear mesh frequency side bands was performed. The calculation was executed by averaging frequency bins over 5 samples, instead of 40 samples, for the area of frequency surrounding the gear mesh fundamental frequency. The total length of each input vector created from each data file was 45.

In order to classify a wider variety of data, the dimensionality of the SOM was increased to 10x9. To properly label the data, the clustering algorithm was ordered to set its dimensionality such that each input vector on the map belonged to a different cluster. Later, the clusters were combined to allow training vector to be classified together if needed (i.e., the two normal training files should be in the same cluster).

Below are the results of the SOM/clustering algorithm for the combination actual/simulated data set. Again, the uniform distance matrix (U-matrix) is displayed on the left.

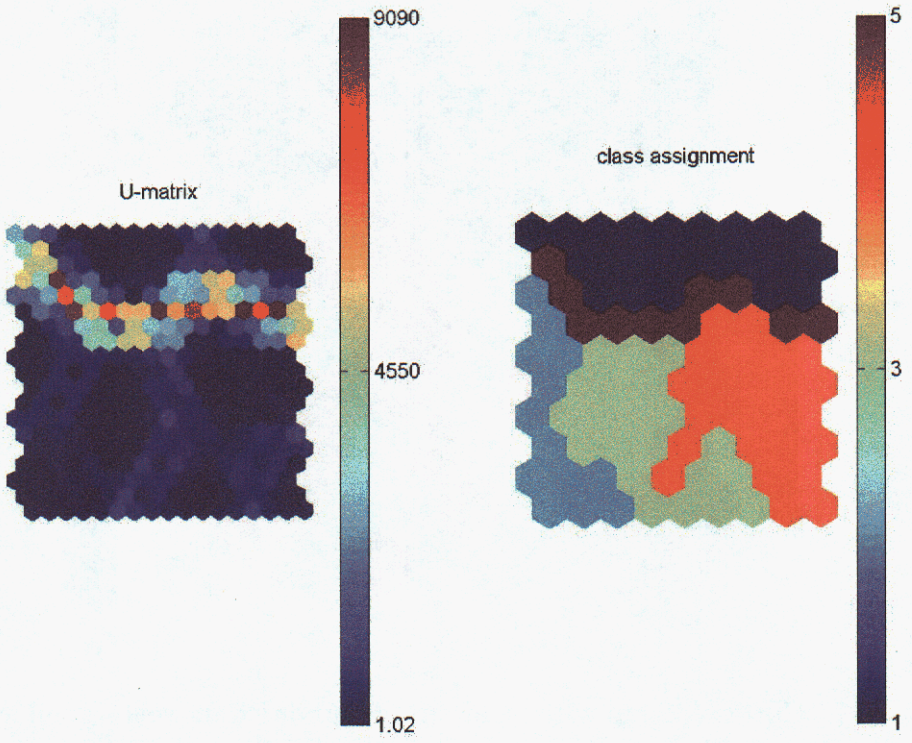


Figure 2.16: U-matrix (left) and output of clustering algorithm (right). Class 1 (dark blue) represents oil-leak classification. Class 2(light blue) represents tooth wear. Class 3 (green) denotes normal data files. Class 4 (orange) represents gear misalignment. Class 5 (brown) is undesignated.

In this case, five different classes were observed, but the data only contained four classes of interest. Class 5 (brown) is undesignated. Compare the brown region on the class assignment map with the same area on the U-matrix. Recall that the U-matrix represents the distance in feature space between adjacent cells. The gear misalignment and tooth wear data were simulated by making slight modifications to normal data files, so one would expect them to be close together in feature space (classes 2,3,4), while the oil leak data (class 1) was actual data. One would expect it to be quite different from the normal data. Hence the active cells on the boundary between the oil leak region, and the region that can be considered as "close to normal." This transitional area results in an extra class, which can be ignored for the purposes of this exercise.

Figure 2.17 displays the locations of the training data on the clustered map. In this case, the algorithm was successful in clustering the training data.

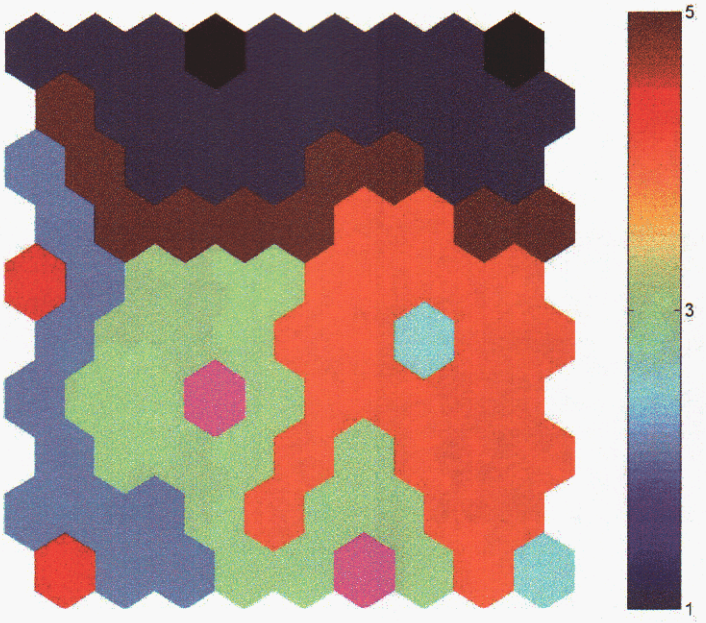


Figure 2.17: Clustered map with locations of training data vectors: oil leak (black), normal (pink), tooth wear (red), gear misalignment (cyan). Compare with figure 2.16.

Next, the trained SOM was given data that it had not seen. The extra data files represented the seven additional oil leak and the four additional normal files described in the previous section, plus simulated misalignment and tooth wear examples created from the four additional normal files. The results of this test are plotted below.

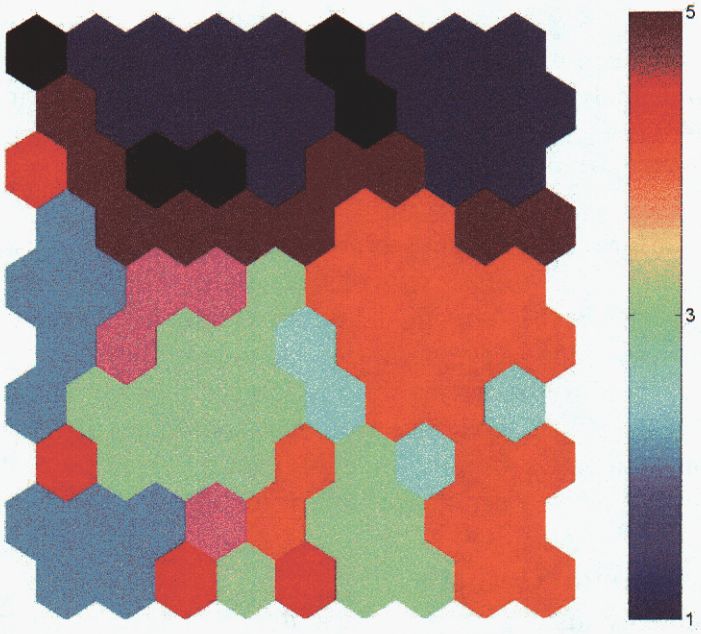


Figure 2.18: Clustered map with locations of testing data vectors: oil leak (black), normal (pink), tooth wear (red), gear misalignment (cyan). Compare with Figure 2.16.

In this test, the algorithm correctly classified all the oil leak data, the normal data, and the gear misalignment data. However, it misdiagnosed one tooth wear file (red cell in bottom, center) as normal, while correctly identifying the remaining three. On the other hand, the network did correctly classify 94% of the testing files it was presented. Eight files containing four different states is a very small amount of data for a SOM to use in training. Such artifacts as misidentification of tooth wear data could likely be remedied with the acquisition of additional training data in the form of additional normal files to manipulate, or ideally, actual tooth wear data. Another solution would be to apply a confidence measurement to map locations, where files near class borders would be issued a lower confidence number.

This exercise demonstrated that a Self Organizing Map could be a useful tool in PHM classification. Although the map did not correctly identify every new data file presented to it, it was able to achieve a near perfect score based on very little training data.

1.7. Vibration Analysis

As part of the Evidence Engine work, we have looked at algorithms for vibration analysis. Vibration analysis is a critical component of prognostics for gearboxes, turbines, motors, etc. There is a wide body of literature on this subject [Wowk, 1991]. A quick summary is given below.

The analysis of sensor readings from vibration sensors involves many signal processing steps: signal conditioning, various types of filtering, enveloping, time synchronous averaging (TSA). TSA involves using data from a tachometer to correctly identify which sample points correspond to a revolution (for example, if the vibration sensor is recording at 10kHz, and the speed of the driving shaft is 250 Hz, then 40 samples would be taken every second. However, due to various problems, sometimes 38 samples may correspond to one revolution, sometimes 41 samples, etc.) TSA involves interpolating and averaging the raw data to obtain a better estimate of the "true" signal.

1.7.1. Time Domain Methods

There have been many signal processing "features" that have been developed over the years. In the time domain, the following statistics of the time domain signal are often used for early detection of bearing and gear damage:

- RMS – Root Mean Square
- Skew – Third moment of the distribution, measures the symmetry of the distribution
- Kurtosis – Fourth moment of the distribution, measures flatness or peakedness of the distribution as compared to a normal.
- Crest Factor – Ratio of the PeakLevel of the signal to RMS.

1.7.2. Frequency Domain Methods

In the frequency domain, the following analysis techniques are used to detect changes in the frequency spectrum that may indicate component failure:

- FFT Fast Fourier Transform. Analyzes the frequency content of the signal.
- Cepstrum Analysis. Measures the intensity of the harmonic content. Is an inverse FFT of the log spectrum of a regular Fourier Transform.
- Enveloping. Used to monitor high-frequency structural responses to periodic impacts such as gear or bearing faults. Involves pre-processing the signal with high or band-pass filter, than using an envelope detector.
- Demodulation. Identifies periodicity and amplitude differences in the modulation of the carrier, such as gear mesh frequencies.
- Interstitial. Involves processing the noise floor data (data between two peak frequencies, such as data between gear mesh harmonics).
- Comblet – Comb filters remove harmonically related noise from signals.
- Wavelet transform – like a FFT, but with a finite basis function

Specific features have been developed for gearbox diagnostics and prognostics. They are listed in Table 2.8 [Society for Machinery Failure Prevention Technology, Gear Diagnostic Parameters]. The Applied Research Laboratory (ARL) at Penn State University has developed a MATLAB toolbox which implements many of these features [Banks and Maynard, 2001].

Table 2.8. Summary of spectral features for vibration analysis of gearboxes

Feature	Method/What it detects
FM0	Detects major changes in gear meshing patterns. Ratio of peak to peak amplitude of TSA and amplitude of gear mesh frequencies.
SLF	Sideband level factor, detects a bent or damaged shaft. Ratio of the first order sideband level divided by standard deviation of TSA.
S01, S02	Detects shaft imbalance or damage. TSA amplitude corresponding to first (second) order shaft frequency.
NA4	Developed to detect onset of damage.
FM4	Detects changes in vibration pattern resulting from damage on limited number of teeth.
M6A, M8A	Detect surface damage
NB4	Detect onset of damage

Many of the features developed for prognostics look at a very small subset or section of the data. We were interested in developing an analysis technique for closely following the progression to failure. Thus, we developed an algorithm to compare the frequency content of any two signals. This algorithm is outlined below.

1.8. Frequency Domain Algorithm for Signal Comparison

To compare sets of vibration data, we have developed an algorithm to statistically compare the frequency content of any two signals. The overall approach allows one to identify differences in magnitudes at particular frequencies, and see what characteristics are changing in the FFT spectrum over time. This algorithm may be useful to detect an impending failure in rotating equipment such as gearboxes. We have designed the algorithm using vibration data from gearboxes, and this application is discussed below. However, since this algorithm can compare frequency content of signals, it potentially can be used in other applications such as voice identity checking and speech recognition.

Current military aircraft gearboxes are manufactured to exacting tolerances and when properly serviced, have a wear-out period that is much longer than that of non-precision equipment such as an automobile transmission. It is very difficult, if not impossible, to accurately trend the slow degradation of such a high-precision piece of equipment. However, we do believe it is possible to detect when the cumulative degradation has

reached a statistically significant level or when a sudden change occurs, such as a broken gear tooth or cracked bearing.

1.8.1. Approach

The approach of this algorithm is to establish a baseline signal under fault-free conditions, establish a comparison signature, and perform hypothesis tests at each resolved frequency in the frequency domain. If one or more frequencies in the comparison signature are statistically significantly different than those in the baseline signature, the algorithm will indicate that there are differences. Graphically, these steps are shown in Figure 2.19. These steps will be explained in the next section.

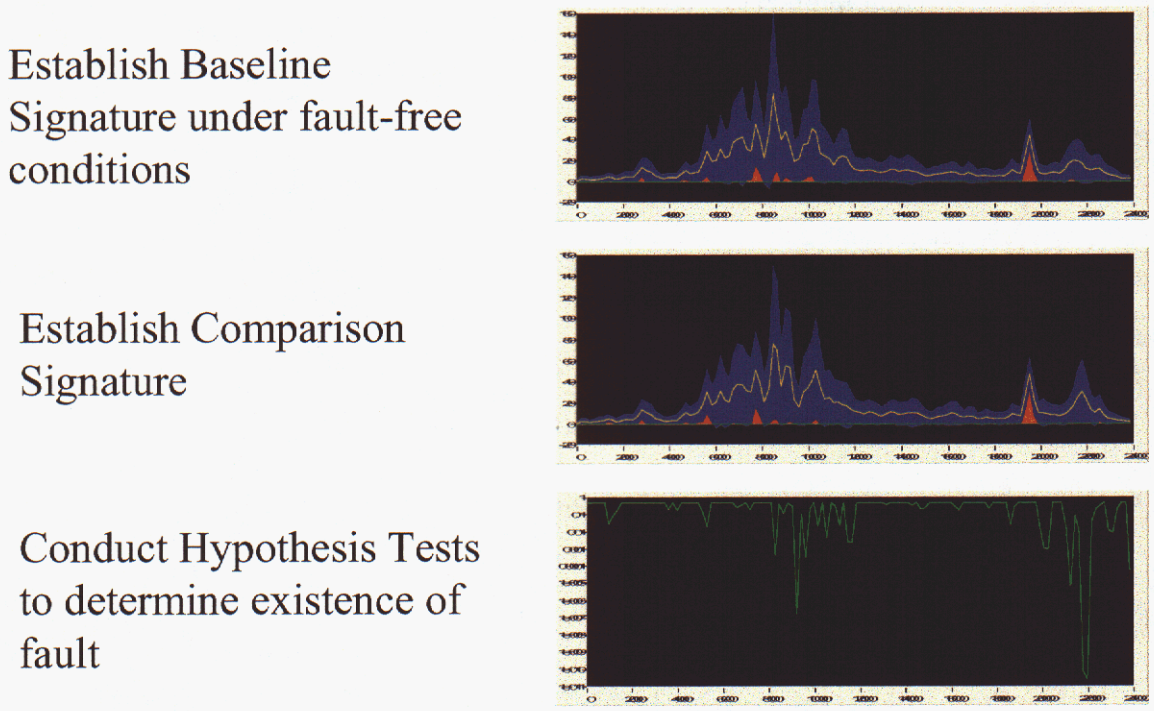


Figure 2.19 Steps in the Frequency Domain Signal Comparison Algorithm

To implement this approach, a vibration sensor must be rigidly mounted on a gearbox (or other piece of equipment) in an ideal location. The optimal placement will be chosen such that all frequencies within the gearbox are well represented.

We initially conjectured based on observation that the autospectrum would be the ideal starting point from which to perform our hypothesis testing. When a gearbox is operating under constant load at a constant speed, we have observed that the auto-spectrum has a very stable plot. Figure 2.20 shows a waterfall plot of the auto-spectrum with time along the z-axis. The data was taken from a Wilcoxon accelerometer magnetically mounted on an accessory drive gearbox during a motoring test. Keep in mind that this particular mounting condition is far from optimal. Nonetheless, the data is extremely consistent. The auto-spectrum is basically an FFT with all the white noise removed without any loss of signal. The auto-spectrum is actually the FFT of the auto-correlation. The auto-correlation is simply the signal multiplied by a time shifted version of itself, where time

is the independent variable. This efficiently removes white noise from the signal while preserving the signal itself.

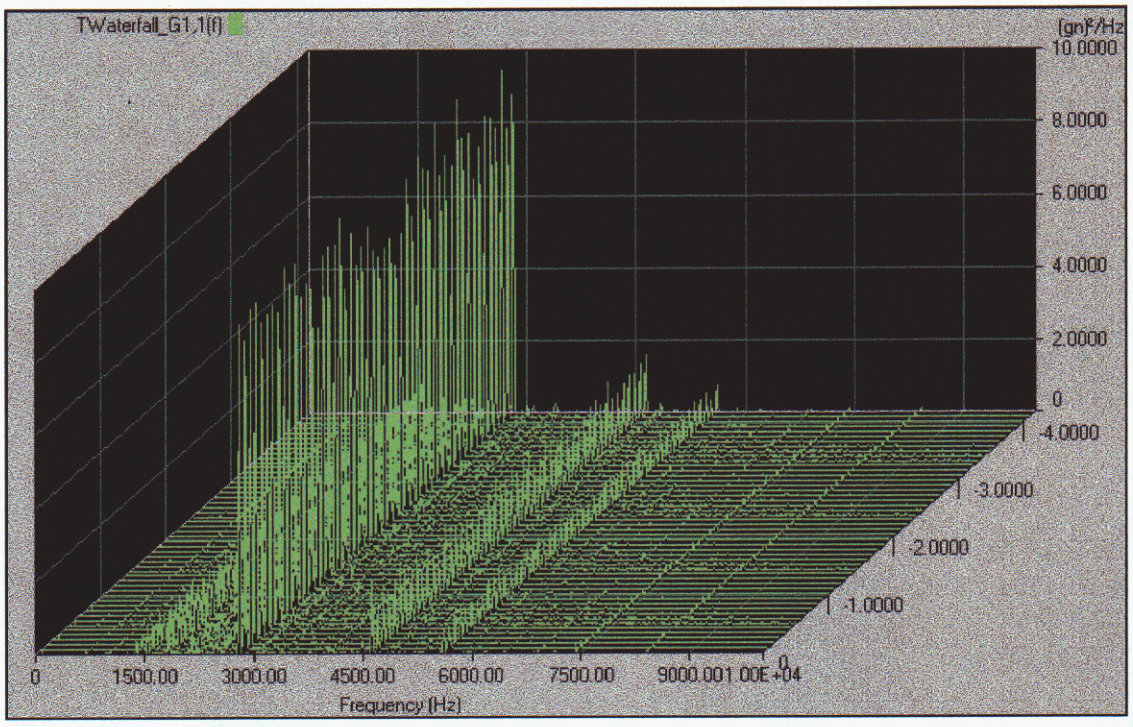


Figure 2.20. Waterfall Plot of Auto-Spectrum

While the autospectrum is more stable than the FFT, a sample of the amplitude values at any given frequency tend to have a skewed distribution. A standard FFT, on the other hand provides amplitude distributions that are much more Gaussian in shape. While the autospectrum does tend to get rid of noise, it is overkill to do so because averaging over many frames already produces the desired effect of averaging out noise.

When one performs an FFT on a signal, the result is a list of complex-valued numbers. At this time, no information has been lost because the FFT is invertible back to the time domain. Ordinarily, we only plot the absolute value of the FFT. We will enhance our ability to detect persistent variations in the signal by averaging over many frames of the frequency spectrum. The only requirement for doing so is that the signal content remains relatively stable during the timeframe of interest. For a gearbox, this means no significant RPM or load changes and no significant transient events occur during the period of measurement.

To baseline a particular gearbox, one would need to collect 100 consecutive frames of the auto-spectrum while the gearbox is in good working order, installed in an aircraft, and running at a defined constant speed and load. Each frame is based on 256 (or a similar power of 2) consecutive samples from the time domain. For each frequency value, we compute summary statistics (i.e. average and standard deviation) of the associated amplitude. We have seen in our gearbox data that the amplitudes appear to be normally distributed, but we can accommodate any sort of distribution once it has been determined.

Currently, it is not possible to establish a baseline signature which applies to all gearboxes. The slightest differences in mounting, non-critical gearbox variations, or the sensor itself would cause significant signal variations among different gearboxes. However, with exceptional quality control on sensors, gearbox manufacturing and sensor installation, it may eventually become possible. While it may be possible to establish the baseline signature at the factory, we currently believe that the best time to establish the baseline signature is after installation on the designated aircraft and before flight testing.

The baseline signature can be defined as a table with three columns: frequency, average, and standard deviation. The table would have 128 rows given 256 time domain samples. Assuming double precision arithmetic, the entire baseline signature would occupy only 3 kilobytes of memory. Because the memory and computation requirements of the algorithm are low, we could create and store a multitude of signatures under a wide variety of operational regimes. The entire signature can be collected and statistics calculated in less than a second.

Once a baseline signature has been established for a particular gearbox, we can repeat the above process to define a signature at any point in the future while the aircraft is running under the prescribed conditions. This new signature is referred to as the "real-time signature".

We then perform a series of hypothesis test on the comparison of two means at each point in the frequency domain, using a very low alpha. The null hypothesis for each test would be that the mean amplitude of the real-time signature is within a specified percentage of the mean amplitude of the baseline signature. Alpha is defined as the risk of falsely rejecting the null hypothesis (a false positive). Alpha must be chosen such that the probability of a false positive for the entire test is very low (<1 in 1000).

Equation 2.1 is the inequality relating the overall false positive probability (p) to alpha. If the tests were statistically independent, the expression would become an equality.

$$p \leq 1-(1-\alpha)^n, \text{ where n is the number of hypothesis tests performed. (2.1)}$$

For example, to have a false positive rate of less than 1 per thousand sorties, assuming that we perform 8192 hypothesis tests in a comparison of real-time versus baseline signature, we need to test at an alpha level of 1.22×10^{-7}. This may seem like an extremely low level, but because we are using 100 samples, the power of the test remains high. For example, suppose at a particular frequency, an average amplitude of 0.5 appears where it had previously been zero. Suppose that in both cases, the standard deviation was also 0.5. At an alpha level of 1.22×10^{-7}, the power of the test on a single frequency would be about 95%, meaning that there is only a 5% chance that we wouldn't detect the anomaly in the gearbox signature. In the event that an impending gearbox failure presented itself in multiple frequencies, the overall sensitivity of the algorithm would be greatly increased.

The primary output of the algorithm is the minimum p-value. Each individual hypothesis test at a given frequency produces a p-value. P-values that fall below the threshold indicate that a statistically and a practically significant change has occurred at a given frequency. The steps shown in Figure 2.19 demonstrate this algorithm. At each resolved frequency in the FFT, the average amplitude is plotted (yellow line on top two figures), and the average amplitude ± two standard deviations is plotted (blue area in the top two figures). The red area has been drawn to show where the lower interval is not zero. At these frequencies, the lower limit is given by the red line. The bottom graph shows the p-value at each resolved frequency. Note that smaller p-values are worse, meaning there is a higher chance the spectra are different. Note at the end of the spectra, around 2200Hz, the p-value is the most negative and the signals look the most different.

The hypothesis testing alone will detect statistically significant differences indicating significant wear or impending failure in the gearbox. Seeded fault testing can be used to obtain estimates of remaining useful life based on the magnitude of the signal differences. In the rare event when a positive identification of impending failure is detected during the aircraft start-up sequence, further inspection may warrant aborting the flight plans. However, a second opinion can quickly be gathered simply by repeating the pre-flight gearbox test sequence. If the second test again shows an impending failure, the flight should be aborted and the gearbox removed from the aircraft for further analysis. If the false positive rate for a single test were set at 1/1000, the chances of two sequential false positives would be roughly 10^{-6}, meaning that a real problem almost certainly exists. Most likely, the problem would be with the gearbox, but it could also be that the sensor or associated cabling goes bad, or that excessive engine vibration is being transferred into the gearbox. All of these scenarios would warrant taking the aircraft out of service.

1.8.2. Implementation

The algorithm has been coded using Visual Basic with a full user-friendly GUI and graphical output and several other features which allow the user to become quickly acquainted with one of more signals, allowing them to make inferences about signal content and differences. Due to the simplicity of the algorithm, it could easily be implemented in hardware. The hardware requirements would be minimal by today's standards due to the low processing and memory storage requirements. Many commercially available Digital Signal Processors (DSPs) could perform the bulk of the number-crunching.

1.8.3. Possible Modifications

There is quite a bit of fine-tuning that we can perform on our algorithm. Everything boils down to a trade-off between sensitivity and specificity of the test. High sensitivity means we have a low false-negative rate, while high specificity means we have a low false-positive rate. Ideally, we would like to tailor the algorithm to achieve an optimal combination of high specificity and sensitivity. Of course the final fine-tuning would depend upon the application requirements.

1.8.3.1. Number of Points in the Frequency Domain

Our initial analysis was performed using 256 points in each frame of the frequency domain. This is a fairly low value which yields coarse resolution in frequency. In general, using a large number of points increases our ability to detect defects. However, because aircraft engine speed can vary from the desired value, we may be forced to use a smaller number of points to maintain the desired level of specificity. One caveat, however, is that by using a smaller number of points, we can test at a higher alpha level, which allows us to compensate somewhat for the loss of sensitivity. We can also input more frames in the same timeframe, which helps both sensitivity and specificity.

1.8.3.2. Number of Frames to Average

In the initial algorithm testing, 100 frames of data were used to establish the distribution of amplitudes at each frequency. At 256 points per frame, it takes less than a second to collect this much information. Statistically, there is only a marginal benefit in using more frames. Depending upon the normality of the distribution, we may be able to reduce the number of frames without any significant effect upon sensitivity and specificity. We looked at some extremely high frequency data and found that we could establish stable signatures with as many as 15,000 frames at the expense of a few seconds of 1GHz processor time.

1.8.3.3. Number of Points at which a Hypothesis Test is Performed

Performing a hypothesis test at every frequency would yield maximum sensitivity and minimal specificity. Through seeded fault testing, we can learn which frequencies are the ideal candidates for testing. One possible approach is to reduce the number of hypothesis tests by targeting the key frequencies of interest along with a systematic sampling of remaining frequencies. The general idea is that we can increase specificity without any appreciable loss of sensitivity.

At this time it appears that the inherent specificity of the test is already high enough to allow us to proceed with the "exhaustive" method of testing at every frequency. This specificity can be tuned by adjusting the statement of the hypothesis test to allow for an acceptable percentage of variability. We've found that with a +/- 10% allowance for variability, we do not produce any false positives on the real gearbox and valve leak data that we have examined. This allowance is a measure of practical significance as opposed to statistical significance. This particular allowance was chosen based on the anectodal evidence presented in [Wowk, 1991]. This reference states that two supposedly identical machines from a given manufacturer often have an overall vibration energy level which can vary by up to a factor of 10. Because an aircraft gearbox is designed and built to the most exacting specifications, with the highest quality components, we made the initial judgment that a 10% change in amplitude at any particular frequency under tightly controlled conditions would indicate an internal problem with the gearbox. Setting the allowance much lower than 10% will not properly allow for amplitude fluctuations due to external noise.

1.8.4. Initial Results

We obtained some vibration data from a gearbox running on a test stand. During the baseline test, an unexpected oil leak occurred. The signal comparison algorithm described above was able to identify that the signals were significantly different. The algorithm detected anomalous signals several minutes before the experienced test stand operator noticed the problem. The p-value from this test as a function of time since oil leak started is shown in Figure 2.21. The baseline signature was established with an FFT using 128 frames, 256 points per frame, and a Hanning window, with the p-value threshold value set at 7.8e-06 to yield overall false positive rate of 1/1000.

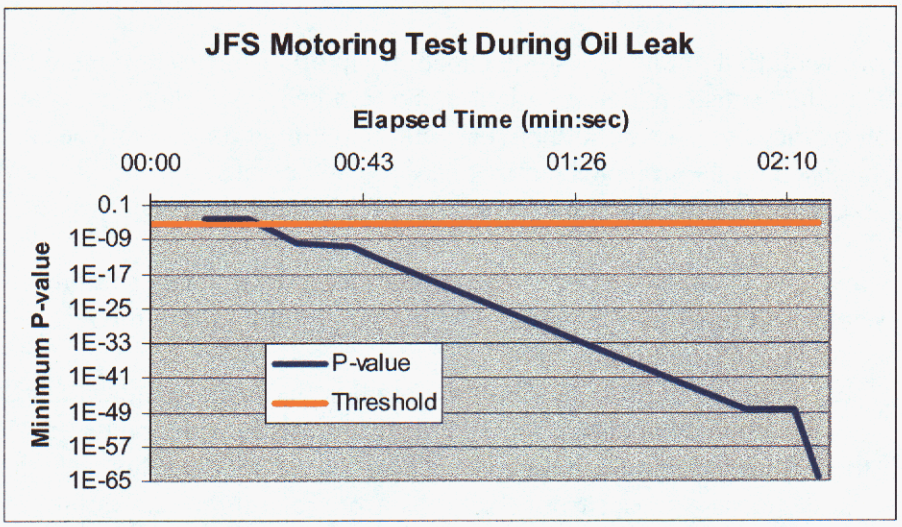


Figure 2.21. Example Results from Signal Comparison Algorithm

To summarize this section, we have developed a frequency domain signal comparison algorithm which looks promising at detecting statistically significant change in amplitudes of frequency spectra. This type of algorithm may prove useful for prognostics of gearboxes and other equipment.

1.9. Evidence Engine Software

We have developed an object-oriented architecture for the Evidence Engine. Algorithms in the Evidence Engine include feature extraction, trend analysis, and Bayesian Belief networks. Objects (major classes) include a Sensor class, a Sensor Feature class, a Distribution class, a Failure Mode class, and a BBN class. Methods within these classes include statistical moments, Fourier transform, polynomial regression, propagation of chance nodes updating a BBN, etc. We have implemented our software in the Visual Basic programming environment, since it is popular tool for object-oriented programming on a Windows platform.

Initially, we want to demonstrate some capability in all of the areas covered by the Evidence engine: sensor feature extraction and trend analysis, data fusion and evidence integration by the Bayesian belief network, and updating a time to failure estimate for a component based on its projected state of health. We created a prototype software program to perform these functions. The goal of the PHM prototype was the following:

1. Allow user to specify a particular failure mode from a list of failure modes on a system, and bring up the parameters relating to that failure mode:
    a. Age of the failure mode
    b. Time To Failure distribution parameters (mean, standard deviation, etc.)
    c. Method of updating the TTF distribution (by sensor data or a BBN)
2. If the failure mode information is updated via sensor data, the user should:
    a. Choose a sensor from a set of sensors relating to that failure mode
    b. Read in sensor data files or sensor feature files relating to the failure mode
    c. If "raw" sensor data is read from files, the user should specify a feature extraction method (such as calculate the mean, variance, skew, RMS, etc. of each batch of data)
    d. Plot a particular batch of data by specifying the date
    e. Plot the feature history of the entire set of data: plot the feature for each batch in the data set.
    f. Specify a threshold where the sensor feature signals imminent failure, and specify a feature extrapolation method (such as linear or higher order least squares) to determine when it is likely the feature will cross the critical threshold.
    g. Use the time between the current time and the time when the feature will cross its critical threshold as an estimate of mean remaining useful life (RUL).
    h. Use this RUL estimate to update the TTF distribution information (go to Step 4).
3. If the failure mode information is updated via a BBN, the user should:
    a. Have the BBN for that failure mode loaded in an "outline" format, where chance nodes and their baseline probabilities are displayed on the screen.
    b. Update the BBN by clicking on the various probabilities and changing them to "0" or "1" to reflect the evidence. For example, if the baseline probabilities for the oil viscosity being a bad state is 0.2 and a good state is

0.8, but the evidence indicates that the oil is bad, the probabilities of being in bad and good states would change to 1 and 0, respectively.

c. Propagate the evidence in the BBN to recalculate the Time To Failure distribution for the failure mode. Update the remaining useful life estimate, and use the RUL estimate to update the TTF distribution (go to Step 4).

4. The user should see the impact of the updated remaining useful life estimates in a concise, graphical format. Specifically, the user should be able to obtain:
   a. The original estimate of remaining useful life and the updated estimate, based on either sensor data or a BBN
   b. The conditional probability of failure X hours from now, given that the failure mode has lasted to the current time and has a Time To Failure distribution that was shifted (reduced mean and in some cases reduced variance) by the evidence.
   c. Probability density functions (PDF) of both the original TTF distribution and the updated TTF.

The following screen shots show Steps 1-4 as implemented in the current prototype version of the Evidence Engine. We expect that these steps and the algorithms underlying them will change, perhaps significantly, if we customize this software for particular applications. However, the current prototype has been designed to easily and quickly examine the impact of different methods and algorithms for updating failure mode health information. We believe it will prove a valuable testbed for that. After the screen shots are presented below, the objects and classes designed in this software will be discussed.

The user starts the application and is presented with a File menu that has standard tasks (open, close, exit). The user clicks on "Open" and is presented with the dialog box shown in Figure 2.22. The user clicks on the file that contains the setup information for this run. In this example, ProgDemo1.txt contains the setup information. The setup information points to two files: a file that contains the list of failure modes for this system and their related information such as TTF distributions and update methods (FM1.txt) and a file that contains the list of sensor data files and BBN data files (FM1Aux.txt).

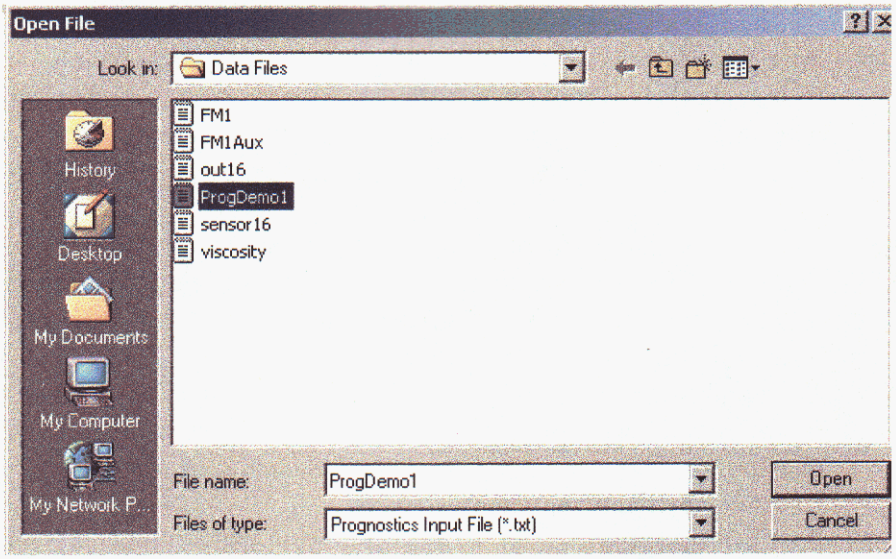


Figure 2.22. Initialization file for Evidence Engine Demo

After the setup file is loaded, a tabbed form with various tabs corresponding to Steps 1-4 outlined above is displayed. The first tab is shown in Figure 2.23. This form presents the user with a drop-down list of failure modes. When the user clicks on one, the information pertaining to its TTF distribution parameters and current age are shown, as well as a PDF of its TTF.

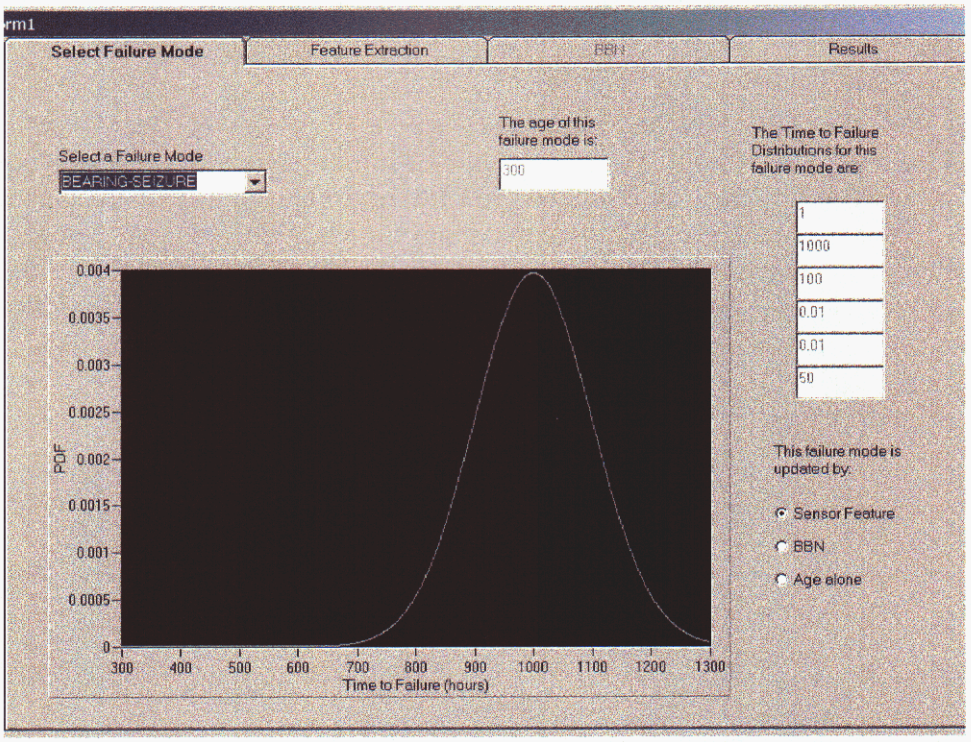


Figure 2.23. Selection of failure mode and display of parameters, PDF

Once the failure mode is selected (Step 1), if it is updated by sensor data, the second tab is entered to obtain sensor data, perform feature extraction, and trend extrapolation (Step 2). This form is shown in Figure 2.24. The user enters the sensor type, the feature extraction method, the critical threshold for that feature, and an extrapolation method for trend analysis (in this case, linear least squares). The user can also select a particular batch date from all the batch data and view that particular batch of data. In the example shown on the form, we have shown the plotting of the feature (in this case, the mean of each batch) over time (green line) as well as the estimate of the feature (blue line). The white line shows the feature threshold and the red diamond shows the date where the feature is projected to hit the critical threshold, in this case, on January 22.

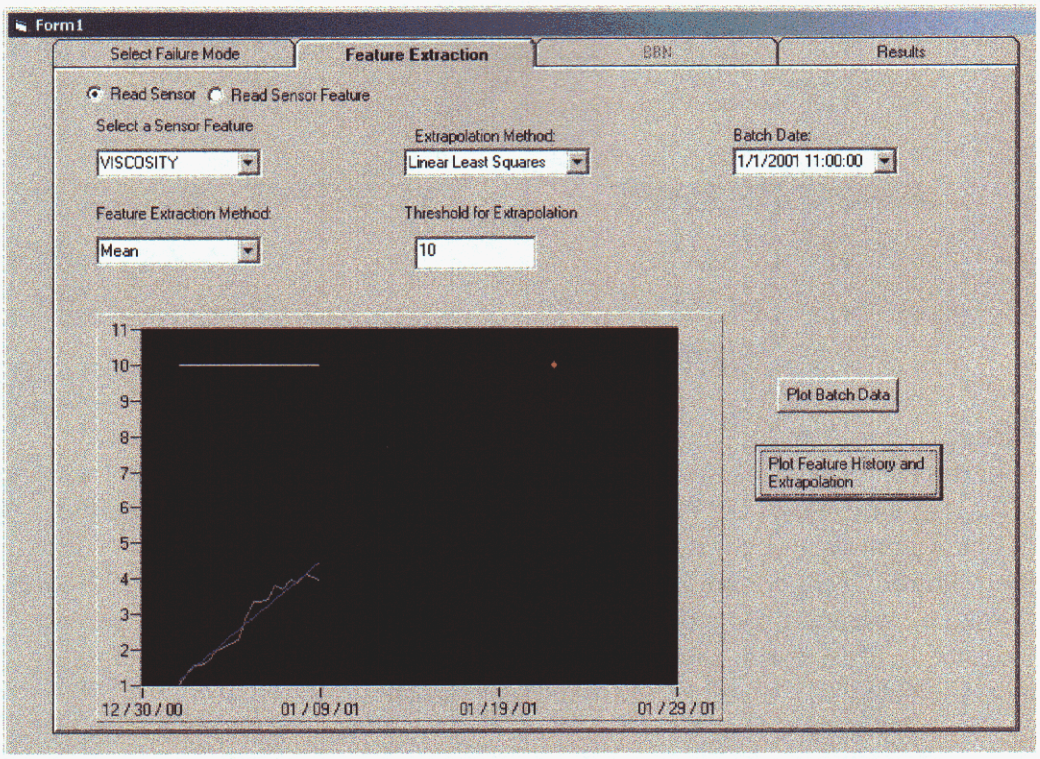


Figure 2.24. Feature Extraction/Trend Projection

If the failure mode is updated by maintenance or inspection information that is modeled by a BBN, the user will enter the tab shown in Figure 2.25 instead of the form in Figure 2.24. Figure 2.25 shows two BBNs: the one on the left side is the BBN with the baseline probabilities, in this case the baseline probability of viscosity, oil pressure, and oil temperature being normal, high, or low. On the right side is the same BBN, but the user can modify the probabilities on this BBN to reflect current inspection evidence. For this example, the evidence indicated the viscosity and pressure were high but temperature was low. These new probabilities are then propagated in the BBN when the user hits the center "Recalculate BBN probabilities" button, and the probabilities of the failure mode are then updated. In this example of lube failure, the initial probability of being in a bad state is 0.05, but that is updated based on this evidence to be nearly 0.58.

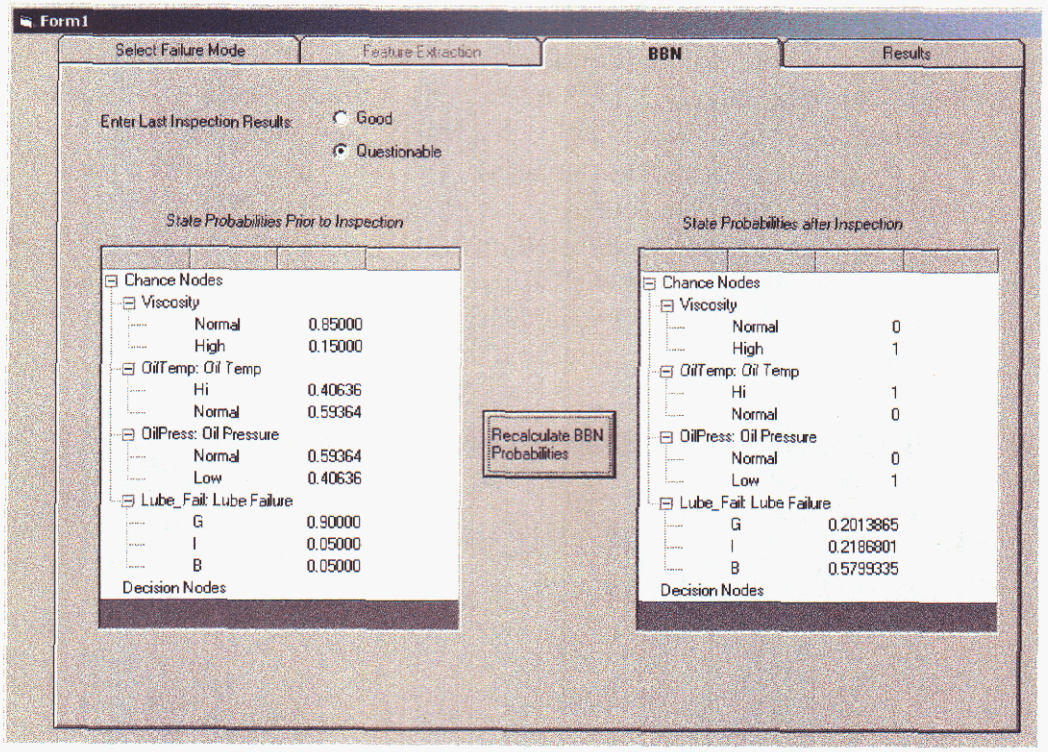


Figure 2.25. BBN Updating

Finally, whether the Failure Mode is updated by sensor data or a BBN, the updated remaining useful life estimate is presented to the user on the last tabbed form as shown in Figure 2.26. This form shows the original and updated RUL. The user can enter the number of hours from the current time at which she wants to obtain an estimate of the probability of failure. This conditional probability of failure (conditioned on the failure mode lasting to the current time and the updated TTF estimates) is given in the box on the lower right side of the form, and the PDFs of both the original TTF distribution (in green) and the updated TTF distribution (in red) are shown.

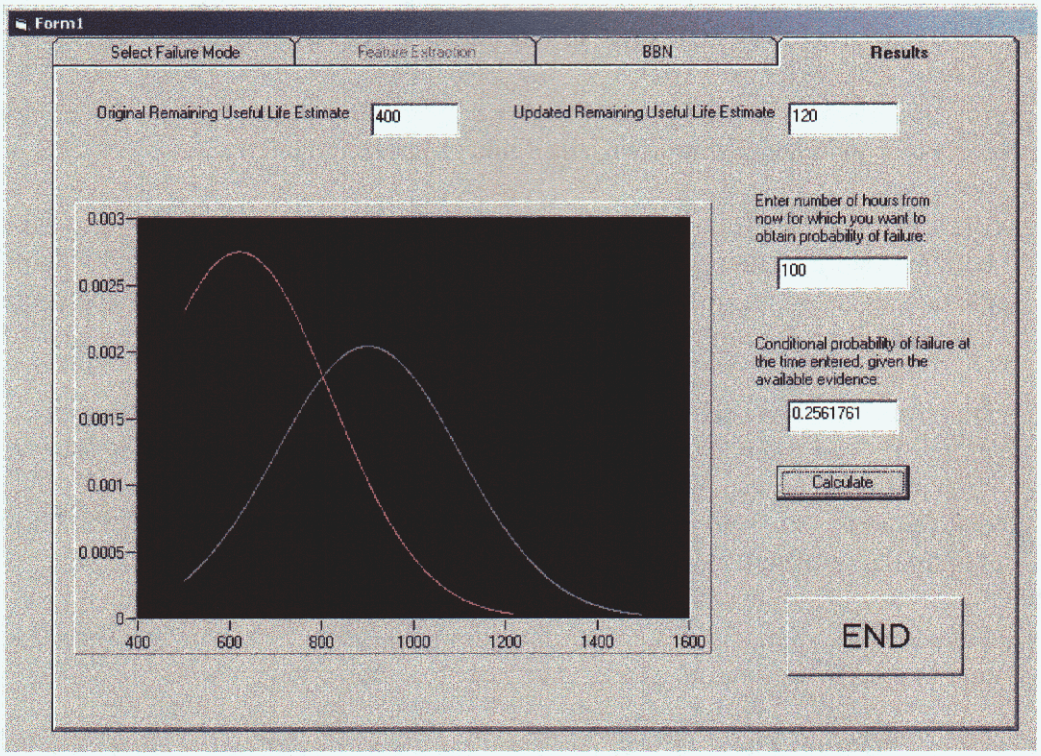


Figure 2.26. Results of updated TTF distribution

1.10. Object Model for the Evidence Engine

The prototype Evidence Engine presented above has been programmed in Visual Basic 6.0. The classes and class hierarchy used in this program include:

- **BBN class.** The BBN class is an object shell for containing a Hugin Bayesian belief network. The class allows one to load a BBN model, compile it, identify chance, decision, or utility nodes, modify the values of these nodes, and propagate updated values through the BBN.

- **Distribution class:** The distribution class is quite large, with a number of properties and methods for calculating cumulative distribution values for a variety of distribution types. The distribution class has an enumerated type of distributions, consisting of the following: Fixed, Wearout, Exponential, Uniform, Triangular, Normal, and Weibull.

- **Sensor class.** Sensors is a collection class for sensor. The sensor class allows one to read in continuous sensor data or batch sensor data.

- **Sensor Feature Class**: The SensorFeatures class is a collection class. The Sensor Feature class allows one to process sensor data, extract sensor features from the input sensor data, and extrapolate or project the sensor feature history to a threshold value. The Sensor Feature class holds both sensor input as well as the feature history, though if the sensor input is in batch form, the sensor feature class only holds the current batch of sensor data since it only processes the feature on one batch at a time. There are many feature extraction methods in this class and we expect to develop more complex feature extraction methods. The current extraction methods are: mean, higher moments (variance, skew, and kurtosis), and RMS. Each extraction method extracts one feature value per sensor batch. For example, if the extraction method is variance and there are 20 batches of data, the feature history will hold 20 variance values. Once the feature history is populated by an extraction method or by directly reading in sensor feature values, extrapolation of that sensor feature to a threshold may be performed. Currently the extrapolation methods are linear or polynomial regression.

- **Failure Mode Class:** The FailureModes class is a collection class. The failure mode class contains information about a failure mode: its ID, Time-to-Failure distribution, the state the failure mode is in (operational, under repair, etc.) The PHM prototype does not use all of the properties of the failure mode class – primarily we use the SensFeatures property to identify the sensor features associated with the failure mode, the TTF distribution, the age of the failure mode, and the update method (is the failure mode health updated via sensor data, a BBN, or age alone?)

- **Combined Distribution Class:** The combined distribution class allows one to input three failure mode state probabilities and associated TTF distributions, and

updates the overall TTF distribution for the failure mode based on an approach explained in the next section. The inputs to this class are the TTFs of the three states, the probabilities of the three states at a particular time, and the conditioning time for each state. The result is a combined TTF distribution (CDF value) and the inverse of the combined CDF value. Because it bears the same properties and functions, the Combined Distribution class is completely interchangeable with the existing Distribution class.

- **BBN Fusion Class.** The BBNFusionClass allows one to take sensor feature values, a sensor feature "partition map", and a Bayesian Belief Network, and output the state membership probabilities of being in various failure mode states. The major methods involved include get/set functions for various nodes in the BBN, an initialization method which reads in the BBN and the feature partition map, and a "fuse" method. The fuse method calculates the state membership probabilities for the sensor features, enters this information as findings in the BBN, then propagates this information to obtain updated beliefs about the state probabilities for the failure modes.

Appendix A outlines in more detail the types of properties and methods that are part of each class.

1.11. Updating Time-to-Failure Distributions

A critical part of the Evidence Engine is updating the Time-to-Failure distributions. A TTF is defined for each failure mode. A failure mode can be in a variety of states. The following approach assumes that the probability of a failure mode being in a particular state has been defined or calculated (for example, from a BBN or SOM).

Suppose we had three TTF probability density functions each associated with a given state (normal, intermediate or severe). We can refer to these functions as $f_i(t)$ and to the states as S_i where i = 1,2, or 3.

Given that a particular component has already attained a certain age, by some means we can compute the Remaining Time to Failure (RTTF) distributions associated with each. (Interestingly, for a function with a constant failure rate the RTTF distribution is no different than the TTF distribution). We can refer to these functions as $r_i(t)$ and the current time as t_{now}. Here is the formula for RTTF for state i:

$$r_i(t) = \frac{f(t + t_{now})}{\int_{t_{now}}^{\infty} f(t)dt}$$

The act of updating a distribution to reflect elapsed time is called "conditioning". Now we have three distributions that actually represent the **conditional probability** that the part will continue to last a specified period of time (with respect to a given failure mode)

given a certain state. A probability for each state is determined by condition monitoring algorithms. Symbolically, we define:

$$s_i := P(State = S_i).$$

To compute the conditional probability that the remaining time to failure falls within the timeframe a<t<b, we would integrate the appropriate density function. In symbols, we have:

$$P(a < RTTF < b \mid State = S_i) = \int_a^b r_i(t)dt$$

We have three possible states, each with its own probability distribution. We essentially have a situation where we know $P(A \mid B_i)$ and the $P(B_i)$ for all i. Thus, we can compute $P(A)$ by the following general formula:

$$P(A) = \sum_{i=1}^{n} P(B_i) \cdot P(A \mid B_i).$$

Applying this to our topic, the unconditional probability that the RTTF lies in the timeframe a<t<b can be computed as follows:

$$P(a < RTTF < b) = \sum_{i=1}^{n} s_i \int_a^b r_i(t)dt$$

Dividing by b-a and taking the limit, we can derive the formula for the overall probability density function of RTTF, which we will call g(t).

$$g(t) = \lim_{b \to a^+} \frac{P(a < t < b)}{b-a} = \lim_{b \to a^-} \frac{\sum_{i=1}^{n} s_i \int_a^b r_i(t)dt}{b-a}$$

$$\therefore g(t) = \sum_{i=1}^{n} s_i \cdot r_i(t)$$

So, perhaps not surprisingly, the correct *a posteriori* distribution under our assumptions is simply a weighted average of the RTTF distributions.

The next logical step was to see this approach in practice with somewhat meaningful distributions. Three empirical distributions were created in Excel and the state probabilities were varied. A wear-out distribution was used for the normal state and exponentially decreasing distributions were used for the intermediate and severe states. Results are shown in Figure 2.27. The cyan curve is the result of weighting each distribution with equal probability. The combined distribution seems to behave in a way that makes sense. As the probability is increased for a given state, while necessarily lowering the probability for the remaining states, the overall distribution assumes the shape of the dominating distribution.

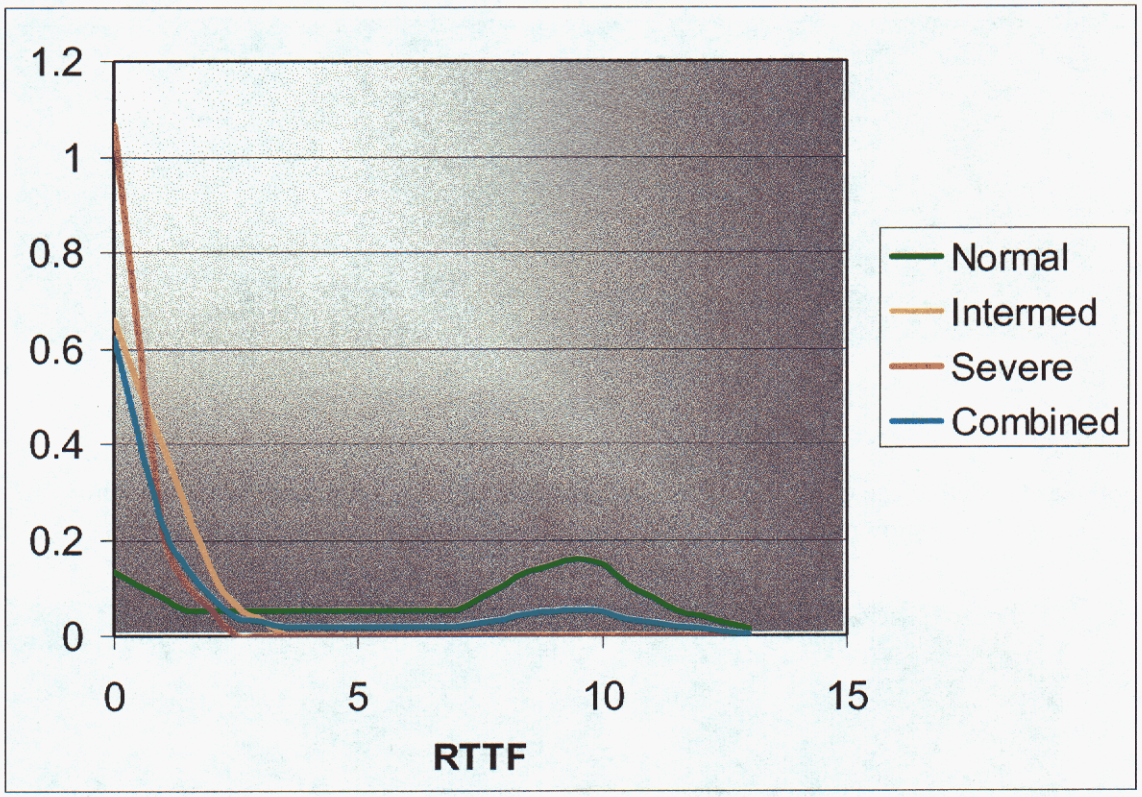


Figure 2.27. Demonstration of Approach showing PDFs for All States

1.11.1. Software Description

In the software implementation, we use the cumulative distribution function (CDF) rather than the PDF described above. Moving the integral inside the sum, the appropriate formula becomes:

$$\therefore G(t) = \sum_{i=1}^{n} s_i \cdot \int_0^t r_i(\tau) d\tau$$

In the algorithm, we also allow this distribution to be conditioned by a specified elapsed time. A detailed description of the "Combined Distribution" class is found in Appendix A.

1.11.2. Examples of Resulting TTFs

We have created an example to demonstrate the TTF updating by combining distributions. Figure 2.28 shows the PDF for a failure-mode exhibiting the wearout distribution while in the normal state

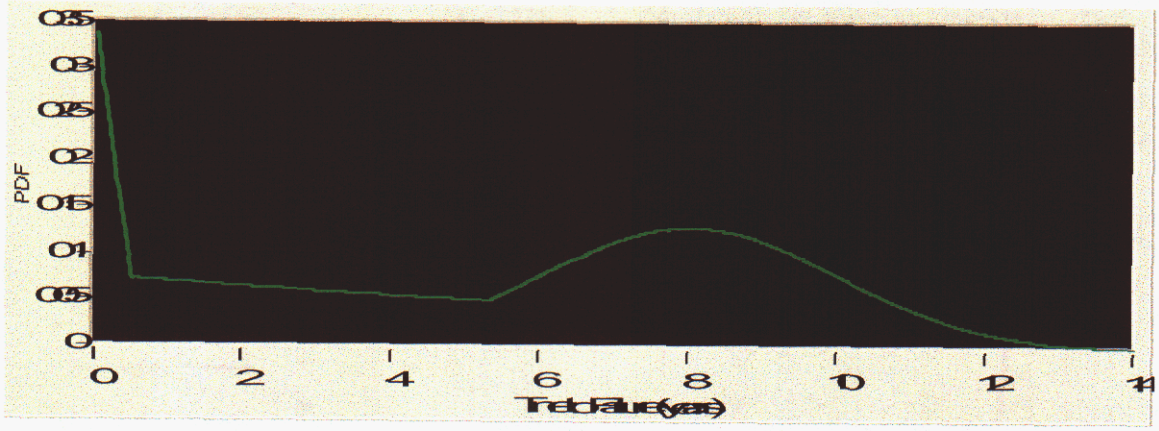


Figure 2.28. Normal State PDF

Let's suppose that the equipment has operated without this failure for 8 years. The resulting PDF is shown in Figure 2.29. The median time to failure is 1.349 years.

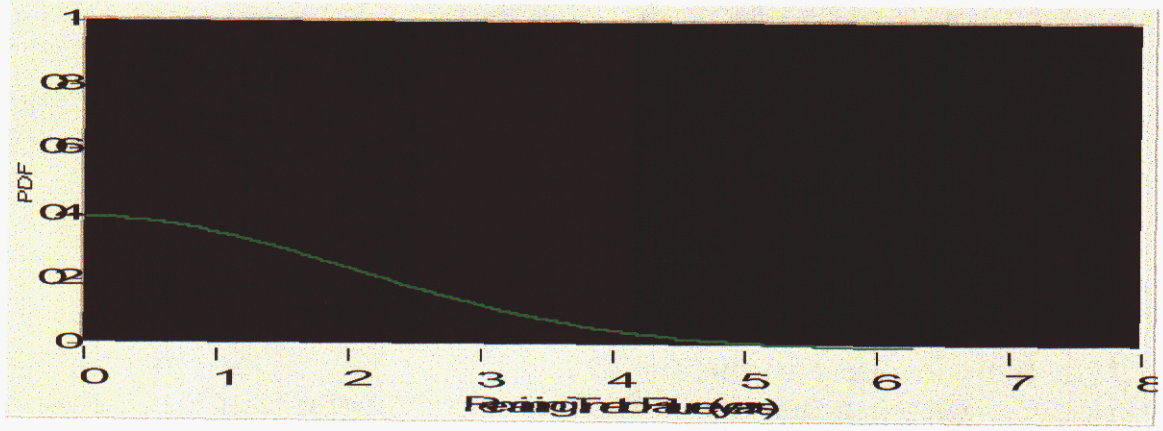


Figure 2.29. Normal State PDF Conditioned 8 Years

Let's suppose that in the intermediate state, the PDF is defined by an exponential distribution with a constant failure rate of 1.0 per year as shown in Figure 2.30.

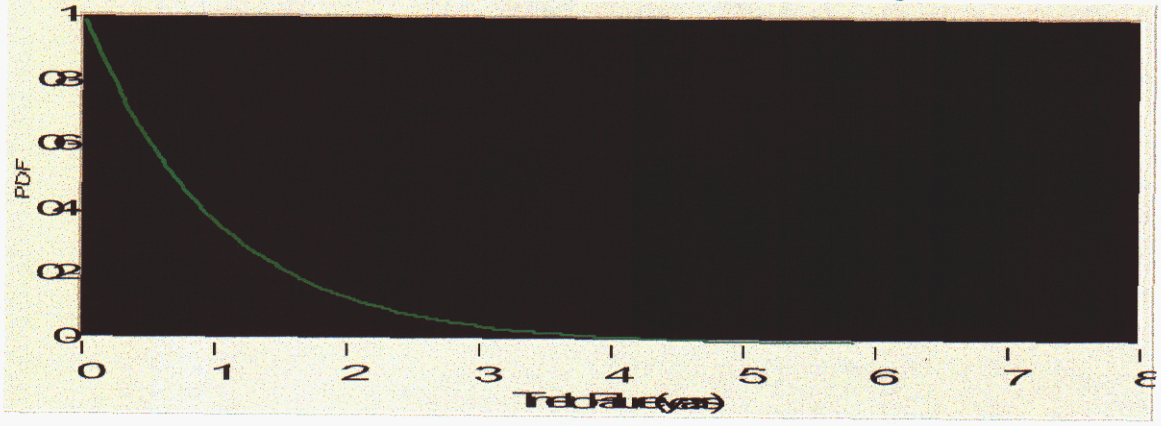


Figure 2.30. Intermediate State PDF

In the severe state, the PDF is defined by an exponential distribution with a constant failure rate of 5.0 per year, shown in Figure 2.31.

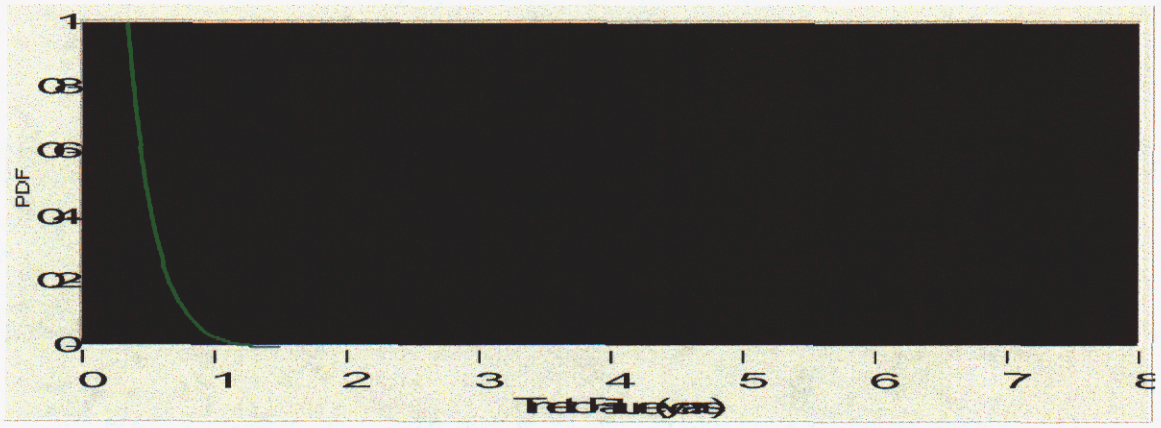


Figure 2.31. Severe State PDF

If we assume equal state probabilities, the result is shown below in Figure 2.32. The median time to failure is 0.494 years.

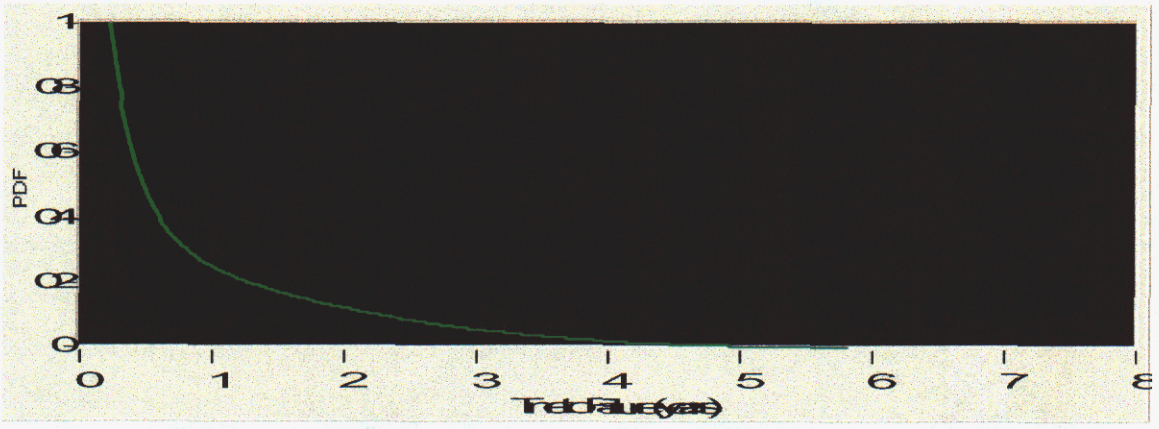


Figure 2.32. Combined State PDF, Probabilities = 0.333, 0.333, 0.333

If we assume state probabilities of 0.1, 0.5, 0.4 for normal, intermediate and severe states, the median time to failure is now 0.361 years (Figure 2.33).

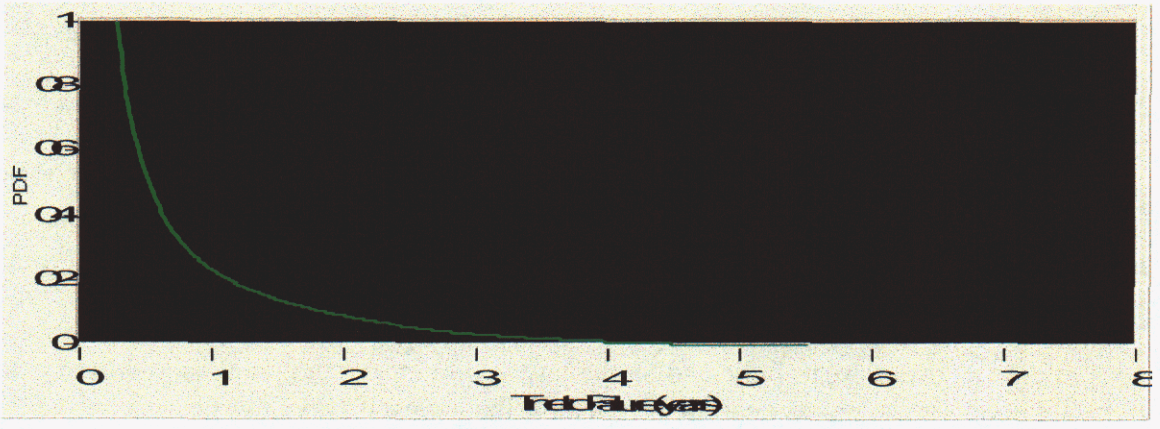


Figure 2.33. Combined State PDF, Probabilities = 0.1, 0.5, 0.4

As the severe state probability increases, the predicted median time to failure decreases further. For example with state probabilities of 0.1, 0.1 and 0.8, we have a median time to failure of 0.181 years (Figure 2.34).

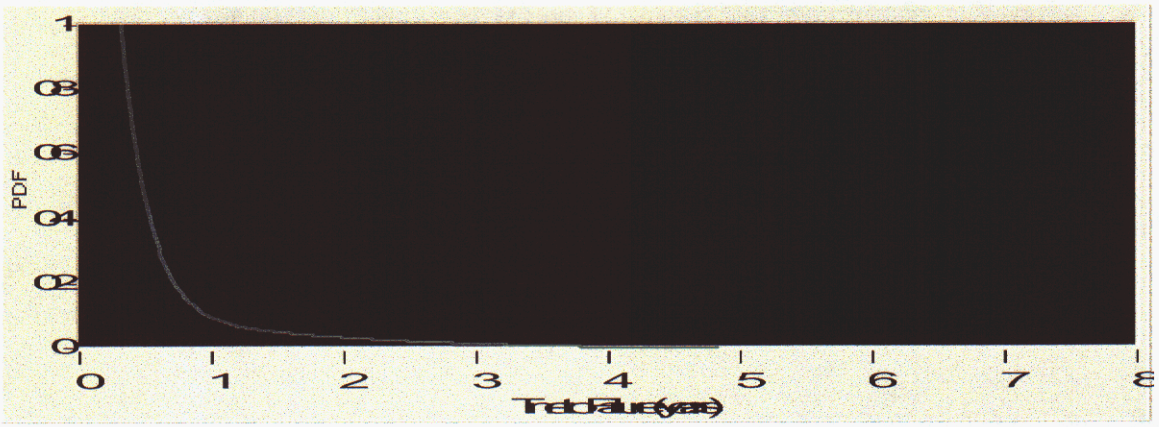
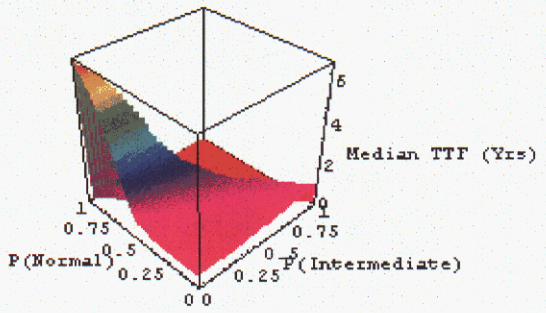


Figure 2.34. Combined State PDF, Probabilities = 0.1, 0.1, 0.8

We can get a better picture of the effect the state probabilities have on the expected median time to failure by looking at a 3D plot of median TTF versus the normal and intermediate state probabilities. In Figure 2.35, the severe state probability is implied, since all three probabilities must sum to unity. The picture below on the left is the result when the wearout distribution is not conditioned. The picture on the right is the result of conditioning the wearout distribution by 8 years.

Normal State Unconditioned



Normal State Conditioned 8 yrs

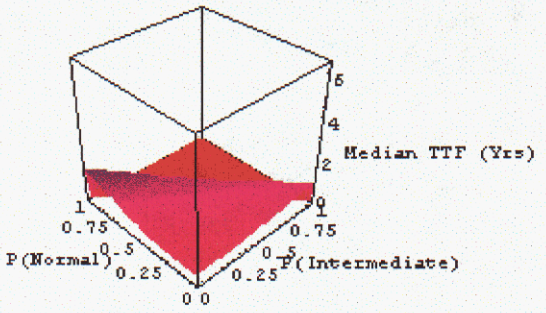


Figure 2.35. Predicted Median TTF versus Normal and Intermediate State Probabilities

Referring to the leftmost graph above, when the normal state distribution is still predicting a lot of life in the component and the normal state probability is high, the predicted median TTF is high. However, as the normal state probability decreases, there is a rather rapid reduction in predicted TTF. Regarding the graph on the right, the normal state distribution has been conditioned by 8 years to reflect the age of the equipment. The intermediate state PDF predicts only a slightly lower TTF than the conditioned normal state distribution, as indicated by the low asymmetry of the graph from left to right. In both graphs, points near the origin have the lowest TTF values because of the high severe state probability.

Consequence Engine

1.12. Introduction

1.12.1. Objectives

This section outlines the design for the Consequence Engine (CE). The purpose of the Consequence Engine is to predict the effect that repair/replace/inspect/wait strategies will have on the system if that action is taken. This "consequence analysis" part of the problem must be performed in such a way that the expected benefit from performing each alternative can be calculated. Then, given a set of alternatives each with an expected cost/benefit, the alternative (or subset of alternatives) can be chosen that maximizes the expected benefit for the minimum cost.

The Consequence Engine supports Prognostic Health Monitoring in two ways: First, when the Evidence Engine identifies a pending equipment failure, it must recommend an appropriate action. Possible actions might be to shut down immediately and repair the problem, ignore the problem and deal with the failure when it occurs, or schedule maintenance at an appropriate time in the future. For the operator or maintenance personnel to make the best decision, they need to know the consequences of all the possible actions. Consequences might be measured in terms of expected outage time or cost. The Consequence Engine provides a capability to quickly evaluate alternate actions and estimate the consequences of each.

The second way in which consequence analysis supports PHM is to provide a capability for rapid evaluation of the potential effectiveness of a PHM system. PHM systems, no matter how well designed, may either fail to detect a pending problem (false negative) or report a problem when none exists (false positive). False positives can result in unnecessary (and expensive) maintenance. False negatives can allow failures to occur that should have been caught. If important decisions are based in part on PHM predictions, the costs of a false negative can be very high. For example, the decision may be made that military equipment is "good to go" for a mission when a pending failure has gone undetected by the PHM system. The Consequence Engine can help understand the cost/benefit trade-offs for a PHM system depending on its error rates.

The outline of this section is as follows: Section 3.2 gives an overview of the Consequence Engine architecture and discussion of the current version. Section 3.3 presents the object models and simulation modules behind the Consequence Engine, Section 3.4 presents a case study example using the Consequence Engine, and Section 3.5 presents results of sensitivity and optimization analyses.

1.13. Overview

The Consequence Engine takes the system health predictions from the Evidence Engine and develops projections into the future: what will be the overall impact on the system if a certain action is taken. Simulation is used to model the long-term impact of possible maintenance/inspection strategies for each failure mode on overall system performance

and cost. We chose simulation as the basis for the Consequence Engine because of its flexibility. The simulation mimics the reliability behavior of equipment in terms of simulated equipment failures, repairs, scheduled maintenance, and inspections. The simulation is based on user-definable maintenance and inspection schedules and a reliability model with time-to-failure and time-to-repair distributions for all failure modes. A spares model is included since the availability of spares may be a major factor in decision-making when a pending failure is identified. By running repeated simulations, the Consequence Engine can be used to calculate performance metrics such as mean time between failures (MTBF), mean time to repair (MTTR), availability, maintenance cost, downtime cost, etc. Thus, by running the simulation with different maintenance schedules, it can be to examine the consequences of alternative equipment maintenance scenarios.

A graphic depicting the architecture of the simulation is shown in Figure 3.1. The simulation integrates and drives several modules: a schedule module, a cost module, a reliability module, and a spares module. The simulation is a discrete-event simulation. It works by examining the schedule of preventive maintenance actions and failure and repair events (events are generated by drawing from Time-to-Failure and Time-to-Repair distributions). A "master clock" looks at each event, calculates the state the system is in given that event, and then finds the next event that will happen. For example, when a failure event occurs, the system goes into a "failed" state. If the repair is scheduled to take 4 hours and no other events happen before that, then the system will return to an "operational" state after 4 hours. If, however, the repair time is 72 hours and a plant shutdown is scheduled 30 hours from now, the simulation will transition the equipment to the appropriate shutdown state 30 hours from now and correctly account for the repair time. At the end of the simulation, statistics on system performance measures such as system MTBF, availability, downtime, and cost are calculated and presented to the user. If desired, the user can view the history of events in the simulation.

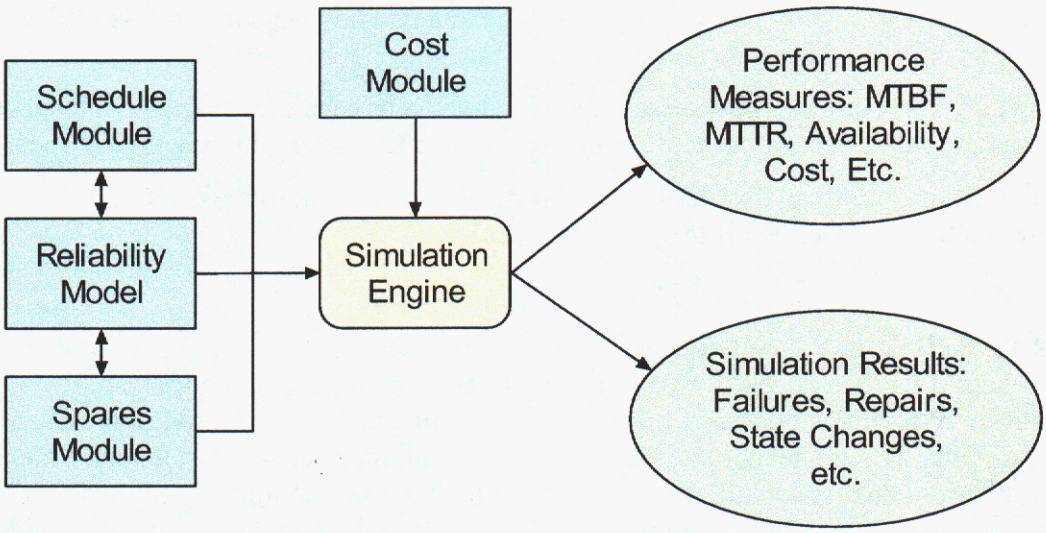


Figure 3.1. Consequence Engine Architecture

1.14. Approach

Figure 3.1 provides an overview of the Consequence Engine architecture and its major modules. This section describes each of the Consequence Engine modules.

1.14.1. Schedule Module

Equipment failures are, by definition, unplanned and are interruptions of the planned use for the equipment. Simulation of equipment reliability behavior can be viewed as creating a chronology of equipment state changes and must begin with the planned use for the equipment. An example of such a chronology is shown in Table 3.1.

Table 3.1. Example Equipment State Chronology

Date	Equipment State	Status
01/01/00 12:00 AM	Operational	Scheduled
01/06/00 09:34 AM	Standby	Scheduled
01/08/00 02:40 PM	Operational	Scheduled
02/10/00 09:11 PM	Failed	Unscheduled
02/11/00 03:42 PM	Operational	Scheduled
04/23/00 08:00 AM	Preventive Maintenance 1	Scheduled

The scheduling module provides the means to specify the planned equipment usage schedule so that component aging, simulated failures, maintenance, etc. can occur in the context of the planned schedule.

Setting up an equipment schedule involves the following steps.

1. Identify *special periods*. Special periods are time intervals during which the equipment is scheduled to be in a state other than its default state. Special periods do not include scheduled maintenance. For example, the equipment may be shut down

for two weeks in July. A start date and time and an end date and time, as well as an equipment state define such intervals.

2. Specify *preventive maintenance (PM)* schedules. For each preventive maintenance activity, the failure modes to be addressed are identified and the schedule is specified.

The details of setting up a schedule, preventive maintenance activities, and the associated failure modes are described in Appendix D.

1.14.2. Reliability Module

The reliability model is based on a collection of equipment failure modes. The possibility of redundancy or non-critical system elements is treated through the use of success paths. The model contains reliability data including time-to-failure and time-to-repair distributions for the failure modes. Details of the failure mode specification are found in Appendix E. The success path concept is described below:

1.14.2.1. Success Paths

A success path is a collection of elements (failure modes, components or subsystems) that, if all are operating, determine the operational state of the system. For example, consider the following simple block diagram model.

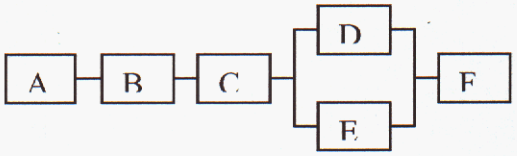


Figure 3.2 Simple Block Diagram Model

In Figure 3.2, elements A, B, C and F are in series while D and E are in parallel. If the functionality of the system in Figure 3.2 is unaffected by whether D or E or both are operating, then two success paths would be needed to characterize the system. They are ABCDF and ABCEF, both of which support full functionality. On the other hand, if the functionality of the system in Figure 3.2 is reduced by the failure of either D or E, then three success paths are needed. Success path ABCDEF supports full functionality while ABCDF and ABCEF support reduced functionality. Of course, series elements do not need to be included in any specific success path since they are, by definition, included in all success paths.

Success paths are defined by a collection of references to failure modes in the reliability model and a reference to the operational state that results from the success path. When a success path is active, the system is in an operational state (note that the user must define these operational states and the groups of failure modes that lead to various operational states).

Operational states may differ from the default operational state in terms of the effect on the system being simulated. For example, a helicopter maneuvering at low altitude is subject to more stress than one flying straight and level at higher altitude. During intervals of increased (or decreased) operational stress, selected components and

subsystems may effectively age more or less rapidly than normal. Alternate operational states provide a means to treat such effects. Operational states are characterized by a collection of affected failure modes and an "acceleration factor" which causes the failure mode to age more or less rapidly than normal during the alternate operational state. For a more complete description of success paths and operational state definitions, see Appendix E.

1.14.3. Spares Module

The spares module treats the spares inventory that is available to the system being simulated. If a failure mode fails during the simulation, the spares inventory is queried to see if a spare part that fixes that component exists, and if it does, how long it take to acquire the spare. The spares inventory is a collection of spare parts each of which has properties such as restock time, withdraw time, purchase cost, storage cost, reorder level, usage rate, etc. Details of the spares module are found in Appendix F.

1.14.4. Cost Module

The cost module assumes that the cost of downtime can be characterized by a function that is piecewise constant. The downtime cost function is characterized by a start date, an end date, and the downtime cost per hour.

In addition to downtime costs, each event (scheduled or unscheduled) can incur a cost. Each failure mode has an optional cost property. Each scheduled maintenance includes a cost to perform the maintenance, which is added to the cost to repair any failure modes addressed by the maintenance. The cost specification is found in Appendix G.

1.14.5. Simulation Engine

A simplified flowchart of the Consequence Engine simulation logic is shown in Figure 3.3. The first step in preparing the simulation is to set up the planned schedule. The first event in the planned schedule is always the simulation itself. The simulation event's start and end dates define the period to be simulated. Other types of events in the planned schedule might include preventive maintenance (PM), scheduled shut down periods, etc.

The Consequence Engine maintains the planned schedule (which is not changed during the simulation), the actual schedule (which is the result of the simulation), and a collection of current events. The actual schedule initially contains only the simulation event: other events are added during the simulation. The current events collection contains events that started prior to the current date but have not yet ended. These events are allowed to overlap.

At any time during the simulation there are three possibilities for the next system state change: 1) the end of a current event, 2) the occurrence of a failure mode, or 3) the beginning of a planned event. The Consequence Engine determines which of these occurs next and takes the appropriate action. If the next event is a failure mode or a planned event, the event is added to the actual schedule and to the current events collection. If the end of a current event causes the next state change, that event is removed from the current events collection. Time is then advanced to the next state change and the process continues. The simulation is complete when no events remain in the current events collection.

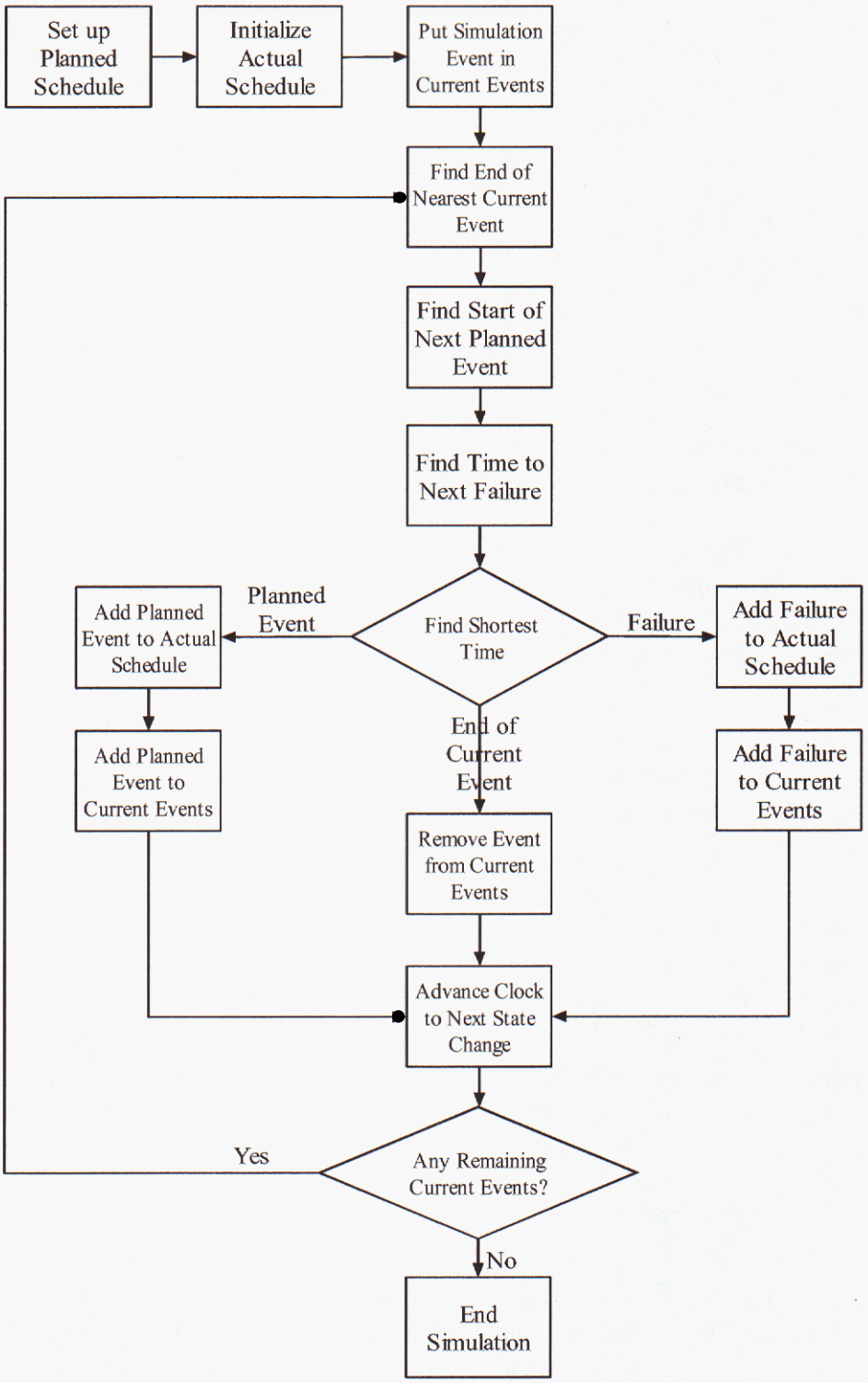


Figure 3.3. Flowchart of the Consequence Engine Simulation Logic

1.14.6. Input and Output Descriptions

The Consequence Engine currently reads input data from a text file and does not provide any on-screen input editing capability. However, a utility program has been developed to allow the Consequence Engine to be used for uncertainty calculations and optimization analysis. The utility program makes it relatively easy to use the Consequence Engine for uncertainty analysis using sampled input data from Sandia's SUNS software or to couple the Consequence Engine to the GO optimization driver. Appendix H describes the baseline Consequence engine input as well as the procedures for using the Consequence Engine in uncertainty and optimization analysis.

In the current version of the Consequence Engine, output results are simply appended to the end of the input file. The results are presented in a way that makes it simple to paste them into a spreadsheet for analysis. Details of the results specification are found in Appendix I.

1.15. Consequence Example

The purpose of this example is to test the reliability simulation capability of the Consequence Engine and to provide a limited validation of its simulation algorithms. The next section presents the example data set and the assumptions required to use the data as Consequence Engine input. Subsequent sections present results of the example analysis and an illustration of the use of the Consequence Engine in scenario evaluation.

1.15.1. Example Input Data

The example application is based on data from pressurized water reactors (PWRs) in the United States during the period of 1990 – 1995 [INEL, 1996]. The applicable data was summarized in an MIT report in 1998 [Golay and Kang, 1998]. This data (Table 3.2) identifies the power plant components most responsible for forced outage time. While the data in Table 3.2 provides the basis for the example application, several assumptions were required before this data could be used. These assumptions are summarized below. In examining these assumptions, keep in mind that the purpose of this application is not to make predictions about the performance of nuclear power plants. The purpose is to provide a test and a limited validation for the Consequence Engine.

1) The outage time attributable to *Other* in Table 3.1 was not presented in reference (2) but was calculated from the percentages.
2) The number of failures attributable to *Other* was assumed to be 100.
3) The number of PWRs operating in the US during the period 1990 - 1995 was 69.
4) Each PWR required four weeks of scheduled outage time every 18 months for refueling.

To proceed with the simulation, it is necessary to know the total operational time of the PWRs represented by the data in Table 3.2. The following equation shows this calculation.

$$\text{Potential Op. Hours Per Plant} = 6\,\text{years} \times 8{,}760\frac{\text{hours}}{\text{year}} + 24\,\text{hours} = 52{,}584\,\text{Hours} \tag{3.1}$$

The above equation accounts for 6 years of data plus an extra day for the leap year (1992). The total potential operating hours for all plants is then

$$\text{Potential Op. Hours All Plants} = 69\,\text{Plants} \times 52{,}584\frac{\text{hours}}{\text{plant}} = 3{,}628{,}296\,\text{Hours} \tag{3.2}$$

To get the actual operating hours, we need to subtract forced and scheduled outage time from the total potential operating hours for all plants. Scheduled outage time can be calculated from assumption 4 above.

$$\text{Scheduled Outages Per Plant} = 6\,\text{Years} \times 12\frac{\text{Months}}{\text{Year}}\frac{1\,\text{Outage}}{18\,\text{Months}} = 4\,\text{Outages Per Plant} \tag{3.3}$$

The total scheduled outage time is then given by

$$\text{Scheduled Outage Time} = 69\,\text{Plants} \times 4\frac{\text{Outages}}{\text{Plant}}\frac{672\,\text{Hours}}{\text{Outage}} = 185{,}472\,\text{Hours} \tag{3.4}$$

The total operational time can be calculated by subtracting scheduled and unscheduled outage time from the total potential operating hours. Scheduled outage time is given in Equation (3.4) and unscheduled outage time is found in Table 3.1.

$$\text{Total Operational Time} = 3{,}628{,}296 - 185{,}472 - 135{,}742.3 = 3{,}307{,}081.7\,\text{hours} \tag{3.5}$$

With the results of Equation (3.5) and information in Table 3.1, we can estimate the MTBF and MTTR for the PWRs represented in the data set.

$$\text{MTBF} = \frac{\text{Total Operational Hours}}{\text{Number of Failures}} = \frac{3{,}307{,}081.3\,\text{hours}}{793\,\text{failures}} = 4170\,\text{hours} \tag{3.6}$$

$$\text{MTTR} = \frac{\text{Total Forced Outage Hours}}{\text{Number of Failures}} = \frac{135{,}742.3\,\text{hours}}{793\,\text{failures}} = 171\,\text{hours} \tag{3.7}$$

Results of the Consequence Engine simulation should compare with the MTBF and MTTR values in Equations (3.6) and (3.7). The results should also reproduce the system/component ranking given in Table 3.2.

Table 3.2. Ranking of PWR Systems/Components Responsible for Forced Outage Time

Rank	System/Component	Outage Time	Percent of Outage	Number of Failures	MTTR
1	Transformer	14,442.20	10.64	39	370.3
2	Main Generator	10,955.30	8.07	70	156.5
3	Turbine	10,654.10	7.85	115	92.6
4	Steam Generator	10,597.60	7.81	46	230.4
5	Reactor Coolant Pump	10,004.10	7.37	47	212.9
6	Service Water System	6,369.50	4.69	6	1061.6
7	Steam Extraction Piping	6,362.80	4.69	4	1590.7
8	Diesel Generator	5,828.10	4.29	12	485.7
9	Control Rod System	4,194.60	3.09	51	82.2
10	Main Feedwater Valve	4,147.40	3.06	60	69.1
11	Pressurizer	4,073.40	3.00	20	203.7
12	Safety Injection System	3,899.40	2.87	8	487.4
13	Reactor Coolant System	3,327.40	2.45	22	151.2
14	Main Steam Valve	3,319.70	2.45	33	100.6
15	Circuit Breaker	3,067.10	2.26	14	219.1
16	Steam Generator Feedpump	2,854.50	2.10	18	158.6
17	Auxiliary Feedwater Pump	2,776.40	2.05	4	694.1
18	Moisture Separator Reheater	2,413.60	1.78	19	127.0
19	Inverter	2,399.80	1.77	12	200.0
20	Condenser	2,185.10	1.61	19	115.0
21	Main Feedwater Pump	1,983.50	1.46	37	53.6
22	Main Steam System	1,225.80	0.90	15	81.7
23	Relay	1,183.70	0.87	12	98.6
24	Intake Valve	1,142.20	0.84	2	571.1
25	Circulating Water	955.40	0.70	8	119.4
26	Other	15,379.60	11.33	100	153.8
	Total	**135,742.30**	**100.00**	**793**	**793**

The following assumptions were used to create the CE input data set:

1. Each of the systems/components in Table 3.2 was treated as a single failure mode.
2. The wearout distribution (see Appendix B) was used for the time-to-failure (TTF) distribution.
3. The burn-in or infant mortality period was assumed to be 1% of the mean life and the probability of failure during the burn-in period was assumed to be 0.5%.
4. The probability of a failure after burn-in but before the onset of the wear out portion of the TTF distribution was assumed to be 0.5%.
5. The mean life of components that reach the wear out portion of the TTF distribution was assumed to be the total operational hours of the system divided by the number of failures for that component.
6. The standard deviation of the wear out portion of the TTF distribution was assumed to be 15% of the mean life.
7. The mean time to repair for each component was calculated as the total outage time attributable to that component divided by the number of failures for that component.
8. The time-to-repair distribution was assumed to be a triangular with the minimum set at 50% of the mean, the most likely equal the mean, and the maximum set to 150% of the mean.

The time-to-failure distribution data was developed from the data in Table 3.2 using the above assumptions. The results are presented in Table 3.3. Time-to-repair distribution data is presented in Table 3.4.

Table 3.3. Time-To-Failure Distribution Data

System / Component	Burn-In Fraction	Burn-In Period	Random Fraction	Mean Life	Standard Deviation
Transformer	0.005	848	0.005	84,797	12,720
Main Generator	0.005	472	0.005	47,244	7,087
Turbine	0.005	288	0.005	28,757	4,314
Steam Generator	0.005	719	0.005	71,893	10,784
Reactor Coolant Pump	0.005	704	0.005	70,363	10,555
Service Water System	0.005	5,512	0.005	551,180	82,677
Steam Extraction Piping	0.005	8,268	0.005	826,770	124,016
Diesel Generator	0.005	2,756	0.005	275,590	41,339
Control Rod System	0.005	648	0.005	64,845	9,727
Main Feedwater Valve	0.005	551	0.005	55,118	8,268
Pressurizer	0.005	1,654	0.005	165,354	24,803
Safety Injection System	0.005	4,134	0.005	413,385	62,008
Reactor Coolant System	0.005	1,503	0.005	150,322	22,548
Main Steam Valve	0.005	1,002	0.005	100,215	15,032
Circuit Breaker	0.005	2,362	0.005	236,220	35,433
Steam Generator Feedpump	0.005	1,837	0.005	183,727	27,559
Auxiliary Feedwater Pump	0.005	8,268	0.005	826,770	124,016
Moisture Separator Reheater	0.005	1,741	0.005	174,057	26,109
Inverter	0.005	2,756	0.005	275,590	41,339
Condenser	0.005	1,741	0.005	174,057	26,109
Main Feedwater Pump	0.005	894	0.005	89,381	13,407
Main Steam System	0.005	2,205	0.005	220,472	33,071
Relay	0.005	2,756	0.005	275,590	41,339
Intake Valve	0.005	16,535	0.005	1,653,541	248,031
Circulating Water	0.005	4,134	0.005	413,385	62,008
Other	0.005	331	0.005	33,071	4,961

Table 3.4. Time-to-Repair Distribution Data

System/Component	Min MTTR	Most Likely	Max MTTR
Transformer	185	370	555
Main Generator	78	157	235
Turbine	46	93	139
Steam Generator	115	230	346
Reactor Coolant Pump	106	213	319
Service Water System	531	1062	1592
Steam Extraction Piping	795	1591	2386
Diesel Generator	243	486	729
Control Rod System	41	82	123
Main Feedwater Valve	35	69	104
Pressurizer	102	204	306
Safety Injection System	244	487	731
Reactor Coolant System	76	151	227
Main Steam Valve	50	101	151
Circuit Breaker	110	219	329
Steam Generator Feedpump	79	159	238
Auxiliary Feedwater Pump	347	694	1041
Moisture Separator Reheater	64	127	191
Inverter	100	200	300
Condenser	58	115	173
Main Feedwater Pump	27	54	80
Main Steam System	41	82	123
Relay	49	99	148
Intake Valve	286	571	857
Circulating Water	60	119	179
Other (assumed 100 failures)	77	154	231

1.15.2. Simulation Results

To ensure adequate statistics, the simulation was run 10 times as long as the PWR operational time represented by the data in Table 3.2. That is, the simulation was run for 4140 operational years versus 414 operational years for the actual data. System level results are shown in Table 3.5.

Table 3.5. Comparison of Simulation Results with PWR Data

Performance Measure	PWR Data	Simulation Results	Percent Difference
MTBF	4170 Hours	4154 Hours	0.4
MTTR	171 Hours	172 Hours	-0.6

For both MTBF and MTTR, the simulation results are within 1% of the original data. We also compare the outage time attributable to each system or component in the analysis. These results are shown in Figure 3.4.

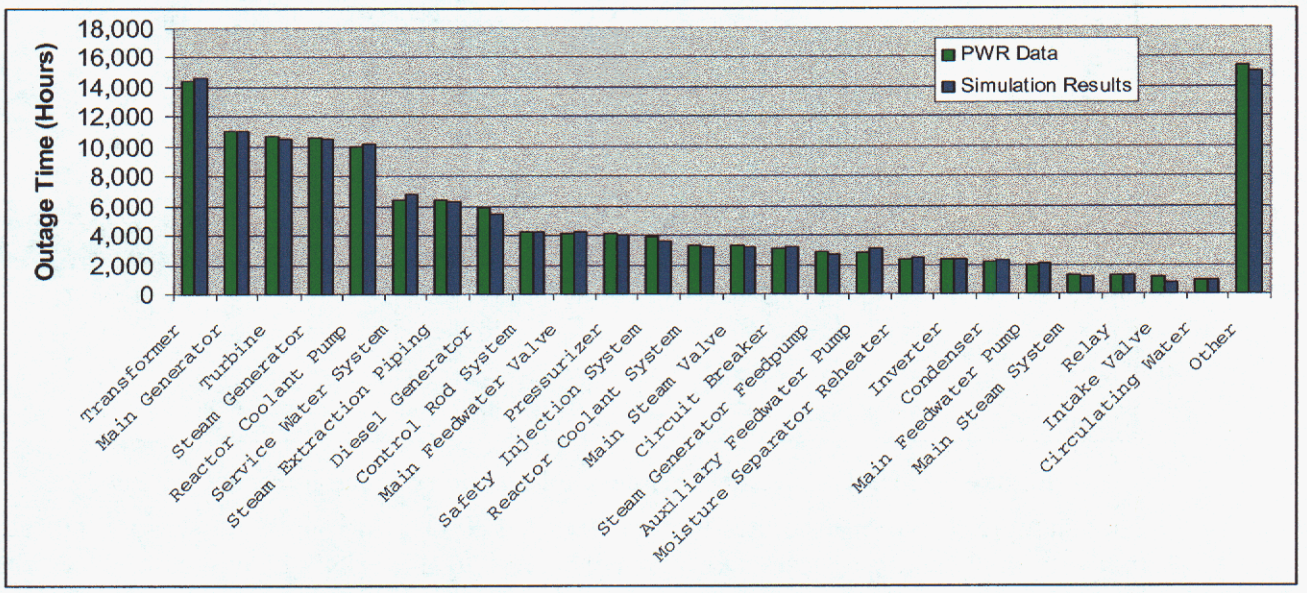


Figure 3.4. Comparison of Outage Times by System/Component

The comparison results presented above demonstrate that the Consequence Engine has accurately recreated the historical reliability behavior of the PWRs represented by the data in Table 3.2.

1.15.3. Example Scenario

This section illustrates the use of the Consequence Engine in evaluating alternative scenarios that could be considered in response to an HMS notification of a pending equipment failure. The basis for selecting the preferred scenario will be the cost of electricity not generated as a result of a scheduled or forced outage.

Cost calculations assume a 1300 MW plant and use a wholesale electricity price forecast provided by Reliadigm, a subsidiary of Public Service Company of New Mexico. Daily average forecast prices for 2002 are shown in Figure 3.5. As shown in the figure, daily average wholesale prices are forecast to range from less than $20/mwh to about $130/mwh. It is clear from Figure 3.5 that the timing of a plant outage can have a dramatic effect on the cost of lost generation.

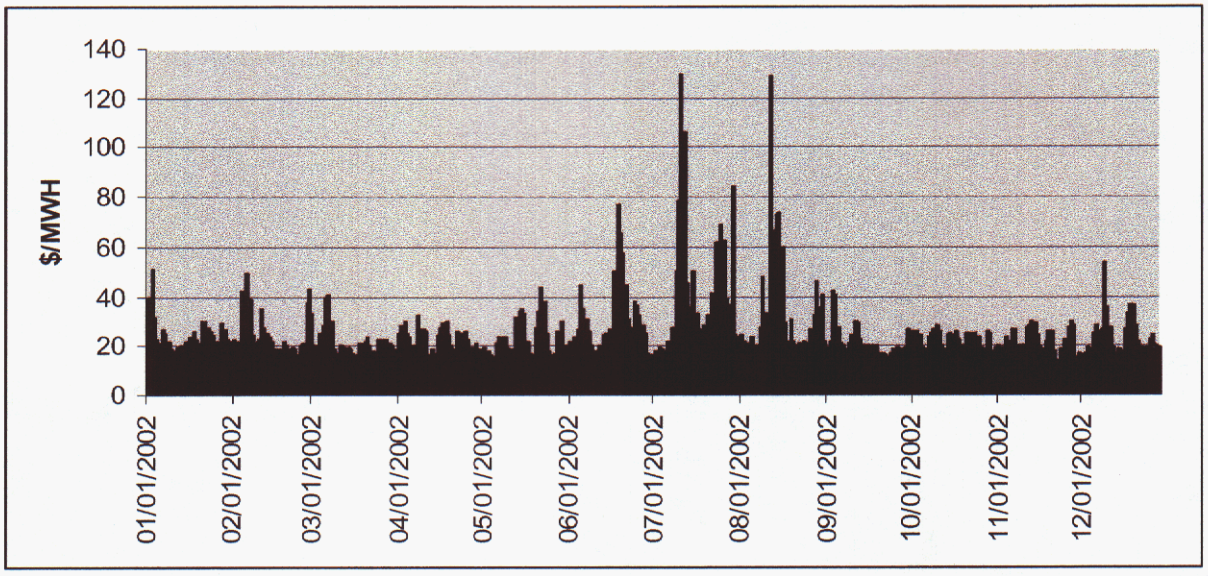


Figure 3.5. Daily Average Wholesale Electricity Price Forecast for 2002

The example scenario postulates that on June 10, 2002, HMS vibration sensors indicate that a turbine failure appears likely to occur in 1 to 2 weeks. In this case, excessive vibration would be expected to cause the turbine to trip. The Consequence Engine was then run for two cases. The first case assumes that we let the turbine run to failure (trip) and the second case assumes that turbine maintenance is scheduled within a day of the warning indication. We assume that the time-to-repair distribution when turbine maintenance is scheduled is the same as the time-to-repair distribution for the run-to-failure case. The two cases are compared based on the cost of lost electricity generation using the wholesale price projections in Figure 3.5. The results for the two cases are shown in Figure 3.6. The expected cost of lost electricity generation for the run-to-failure scenario is about $6.7M whereas the expect cost when maintenance is scheduled immediately is about $2.9M.

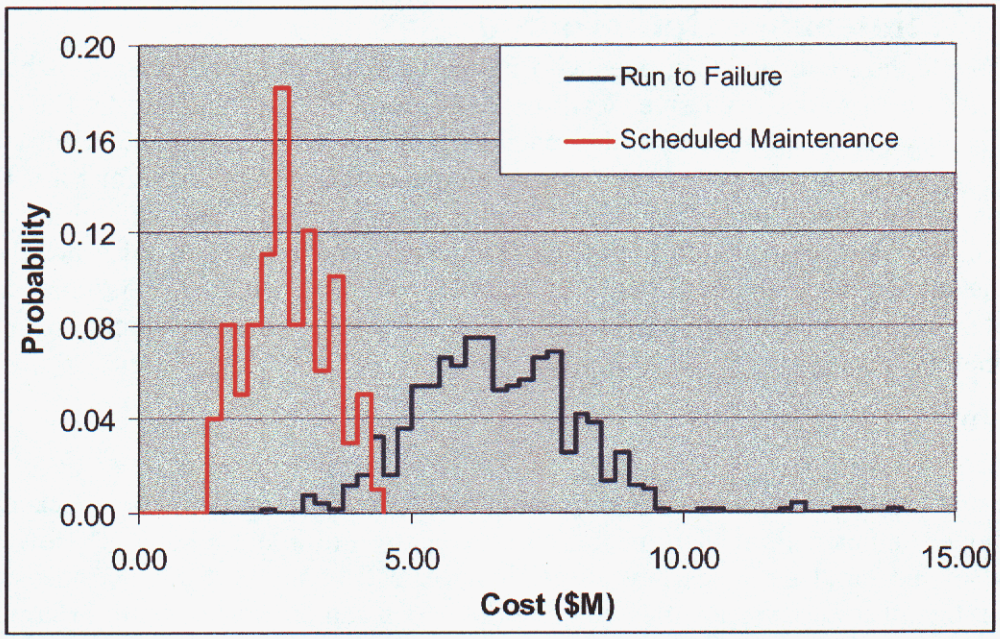


Figure 3.6. Cost Comparison for Two Maintenance Scenarios

To illustrate the effect of maintenance timing on the cost of lost electricity generation, a series of calculations was performed in which the timing of the scheduled maintenance was varied. Specifically, the maintenance was scheduled from 1 to 7 days after the HMS indicating that a turbine failure was expected. Results are shown in Figure 3.7.

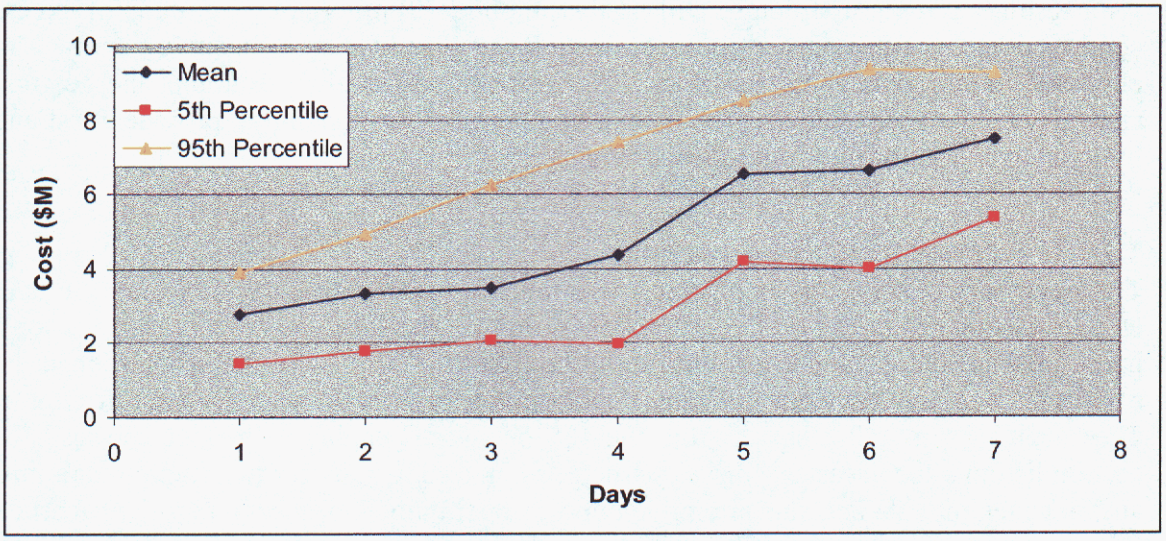


Figure 3.7. Cost of Lost Electricity Generation vs Days Delay in Maintenance

The increase in cost as a function of the delay in scheduling maintenance is a result of the projected increasing wholesale electricity cost for the next few days after the HMS warning. If the warning should occur at a time when electricity cost was projected to decline, it might be more cost effective to delay maintenance subject to the expected timing of the failure.

1.16. Maintenance Optimization

To illustrate the use of the Consequence Engine (CE), an optimization analysis has been performed on an Accessory Drive Gearbox (ADG) for a fixed wing aircraft. The purpose of the analysis is to optimize both the scheduling of maintenance/inspection activities as well as the spares inventory that supports ADG maintenance. Optimization variables for the spares inventory include reorder levels, restock times, and withdrawal times. Maintenance variables include maintenance intervals and inspection false positive and false negative rates. In total, some 65 variables were considered. Assume that the variables are discretized into an average of 10 levels each. Then the number of possible combinations is

$$N_C = 10^{65}.$$

While this creates a rather large optimization problem, it can certainly be attacked using evolutionary algorithms if there is sufficient structure in the output space. However, if several of the input variables have little or no effect on the performance measures of interest, then the output space may be quite flat which can create a difficult or impossible optimization problem. For example, suppose that as many as 30 of the input variables have negligible effect on problem output. Then the true problem size should be

$$N_C = 10^{35}.$$

As a result, any output structure will be lost in about 30 orders of magnitude of excess combinations. For this reason, the optimization was preceded by a sensitivity analysis to eliminate unimportant input variables. As with the power plant example, the gearbox analysis is only meant to be representative of a complex system and to provide a test and limited validation for the Consequence Engine.

1.16.1. Accessory Drive Gearbox

The Accessory Drive Gearbox performs two functions on the aircraft. The first is a main engine starting capability on the ground or in flight. The ADG has mounting pads for a hydraulic start motor, and a gas turbine Jet Fuel Starter (JFS). On the ground, or in the event of an engine flameout in-flight, ADG provides power to start the aircraft engine. The second is providing power from the JFS or aircraft engine to the accessories. To accomplish this the accessory drive gearbox has mounting pads for two hydraulic pumps and a generator. When the aircraft engine is operating, the accessory drive gearbox transfers power from the aircraft engine through the power take-off shaft to the hydraulic pumps and the generator. The ADG contains 29 gears, four lube pumps, one torque converter, three clutches, and other parts. It is a complex piece of equipment (see Figure 3.8)

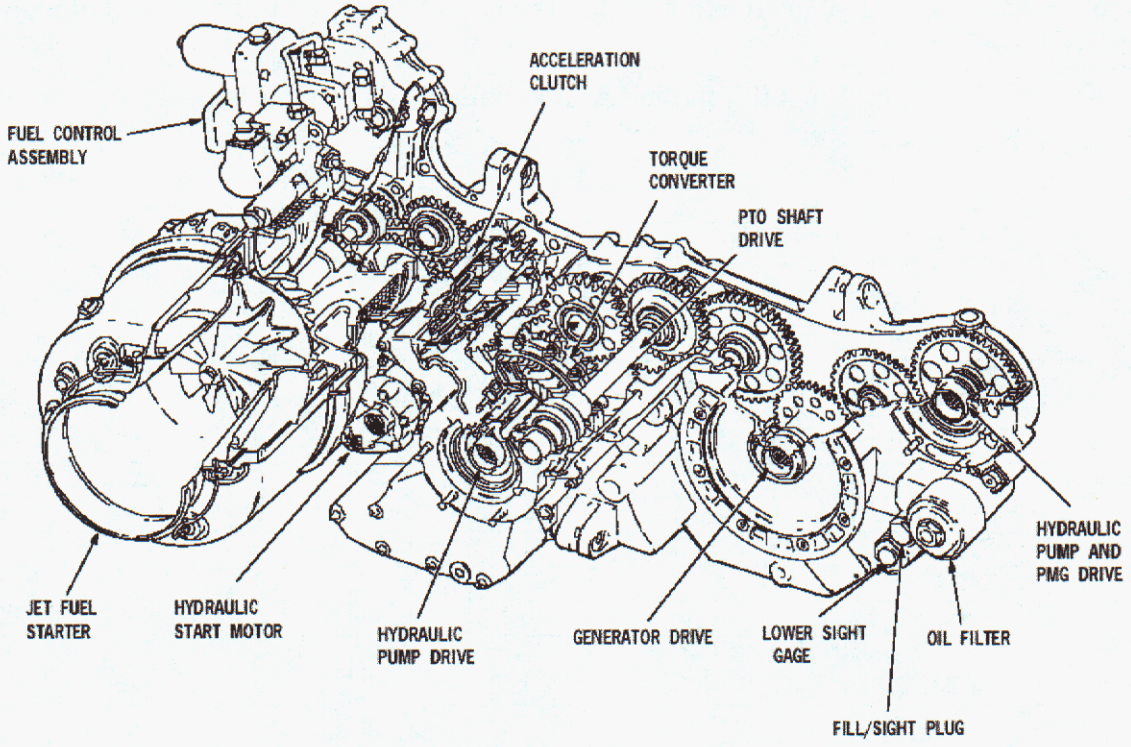


Figure 3.8 Accessory Drive Gearbox (ADG) for a Fixed-Wing Aircraft

For the consequence engine simulation, time-to-failure (TTF) and time-to-repair (TTR) distributions are needed for each failure mode. The wearout distribution, developed at Sandia, was used to characterize TTF for all failure modes. The "Wear-Out" is a three-part distribution developed for use as a time-to-failure distribution. Its three parts are burn-in, normal life, and end-of-life. During the burn-in period, the failure rate is assumed to be linearly decreasing. During the normal life, a constant failure rate is assumed. In the end-of-life portion of the distribution, the time-to-failure distribution is assumed to be normal. The wear-out distribution requires five parameters as follows:

Burn-In Fraction
This parameter determines the fraction of failures that occur during the burn-in period.

Burn-In Duration
This parameter sets the duration of the burn-in period. Its units are hours.

Random Fraction
This parameter sets the fraction of failures that are assumed to occur during the component's normal life.

Mean
This is the mean of the normally distributed end-of-life portion of the distribution. Its units are hours.

Standard Deviation
This is the standard deviation in hours of the normally distributed, end-of-life portion of the distribution.

Table 3.6 shows the TTF distribution for the 15 failure modes that were considered in the analysis.

Table 3.6. ADG Failure Modes with TTF Distributions

Fail Mode	Description	Mean Life	Std. Dev.	Burn-in Fraction	Burn-in Duration	Random Fraction
EAO	ADG TCI	7,530	1,000	0.1	377	0.05
EAA	PUMP ASSY, LUB, ADG	153,000	10,000	0.1	7,650	0.05
EAB	PUMP ASSY, LUB, JFS	27,720	5,000	0.1	1,386	0.05
EAC	FILTER, OIL	13,890	3,000	0.1	695	0.05
EAD	INDICATOR, OIL, DELTA P	10,200	2,500	0.1	510	0.05
EAE	CONVERTER ASSEMBLY, TORQUE	10,000	2,500	0.1	500	0.05
EAF	CLUTCH ASSY, SLIP	22,590	5,000	0.1	1,130	0.05
EAG	VALVE, SHUTOFF, OIL	32,100	5,000	0.1	1,605	0.05
EAH	VALVE, SERVO - ADG CLUTCH	4,050	1,000	0.1	203	0.05
EAJ	PLUG, CHIP DETECTOR	40,500	5,000	0.1	2,025	0.05
EAK	SENSOR, SPEED, JFS	32,100	5,000	0.1	1,605	0.05
EAL	SENSOR, SPEED, PTO	67,200	6,000	0.1	3,360	0.05
EAM	VALVE, SOLENOID	46,800	5,000	0.1	2,340	0.05
EAP	CHECK VALVE, AIR BLEEDER, ADG	100,200	10,000	0.1	5,010	0.05
EBA	SHAFT, POWER TAKEOFF (TCI)	1,500	100	0.1	75	0.05

Parameters for the TTF distributions were developed by first estimating the mean life and standard deviation for components that do not experience burn-in or random failures. Then it was assumed that each component had a 10% chance of failing during burn-in and a 5% chance of failing randomly following burn-in but before the onset of end-of-life. Finally, the burn-in duration was assumed to be 5% of the mean life.

Table 3.7 shows downtime distributions for the 15 component failure modes used in the analysis. The downtimes are assumed to be normally distributed with the means and standard deviations shown in Table 3.7. In each case, the mean TTR includes an assumed 3-hour administrative time in addition to the actual time to repair the failed component. All times are in hours in Tables 3.6 and 3.7.

Table 3.7. ADG Failure Modes with TTR Distributions

Fail Mode	Description	Mean TTR	Std. Dev.
EAO	ADG TCI	33	10
EAA	PUMP ASSY, LUB, ADG	8	1
EAB	PUMP ASSY, LUB, JFS	8	1
EAC	FILTER, OIL	4	0.5
EAD	INDICATOR, OIL, DELTA P	4	0.5
EAE	CONVERTER ASSEMBLY, TORQUE	11	1.5
EAF	CLUTCH ASSY, SLIP	11	1.5
EAG	VALVE, SHUTOFF, OIL	4	0.5
EAH	VALVE, SERVO - ADG CLUTCH	5	0.5
EAJ	PLUG, CHIP DETECTOR	4	0.5
EAK	SENSOR, SPEED, JFS	4	0.5
EAL	SENSOR, SPEED, PTO	4	0.5
EAM	VALVE, SOLENOID	4	0.5
EAP	CHECK VALVE, AIR BLEEDER, ADG	4	0.5
EBA	SHAFT, POWER TAKEOFF (TCI)	7	1

Table 3.8 shows cost information used in the optimization analysis. The non-parts cost is primarily labor but may include testing and other costs. The spares cost column shows the cost to purchase spares needed for the repair. The scheduled repair cost is just the sum of the non-parts and spares costs and is used when the maintenance is scheduled. The unscheduled-repair cost column shows the cost when the component actually fails. In this case, the cost is assumed to be double the cost of doing the same maintenance before the failure actually occurs. The extra costs are intended to account for additional damage that is often incurred when a part fails.

Table 3. 8 ADG Failure Modes with Cost Data

Fail Mode	Description	Non-Parts Cost	Spares Cost	Sched. Repair Cost	Unsched. Repair Cost
EAO	ADG TCI	3,000	40,000	43,000	86,000
EAA	PUMP ASSY, LUB, ADG	500	2,000	2,500	5,000
EAB	PUMP ASSY, LUB, JFS	500	2,000	2,500	5,000
EAC	FILTER, OIL	100	300	400	800
EAD	INDICATOR, OIL, DELTA P	100	800	900	1,800
EAE	CONVERTER ASSEMBLY, TORQUE	800	2,000	2,800	5,600
EAF	CLUTCH ASSY, SLIP	800	2,500	3300	6,600
EAG	VALVE, SHUTOFF, OIL	100	200	300	600
EAH	VALVE, SERVO - ADG CLUTCH	200	1,500	1,700	3,400
EAJ	PLUG, CHIP DETECTOR	100	200	300	600
EAK	SENSOR, SPEED, JFS	100	400	500	1,000
EAL	SENSOR, SPEED, PTO	100	400	500	1,000
EAM	VALVE, SOLENOID	100	300	400	800
EAP	CHECK VALVE, AIR BLEEDER, ADG	100	300	400	800
EBA	SHAFT, POWER TAKEOFF (TCI)	400	10,000	10,400	20,800

The following two tables show data used to model the spares inventory in the analysis. Table 3.9 shows spares purchase, holding, and shipping costs. The purchase cost is the spares cost repeated from Table 3.8. The normal shipping cost is the cost incurred when a replacement part is added to the inventory with normal shipping. When a part is needed that is not in the inventory, an emergency order is placed which incurs the emergency shipping cost in Table 3.9. Finally, the annual holding cost is assumed to be 10% of the purchase cost and is intended to include storage and administrative costs as well as time-value of money.

Table 3.9. Parts Cost Data

Fail Mode	Description	Purchase Cost	Normal Shipping Cost	Emergency Shipping Cost	Annual Holding Cost
EAO	ADG TCI	40000	500	1000	4000
EAA	PUMP ASSY, LUB, ADG	2000	50	100	200
EAB	PUMP ASSY, LUB, JFS	2000	50	100	200
EAC	FILTER, OIL	300	20	40	30
EAD	INDICATOR, OIL, DELTA P	800	20	40	80
EAE	CONVERTER ASSEMBLY, TORQUE	2000	50	100	200
EAF	CLUTCH ASSY, SLIP	2500	50	100	250
EAG	VALVE, SHUTOFF, OIL	200	20	40	20
EAH	VALVE, SERVO - ADG CLUTCH	1500	50	100	150
EAJ	PLUG, CHIP DETECTOR	200	20	40	20
EAK	SENSOR, SPEED, JFS	400	20	40	40
EAL	SENSOR, SPEED, PTO	400	20	40	40
EAM	VALVE, SOLENOID	300	20	40	30
EAP	CHECK VALVE, AIR BLEEDER, ADG	300	20	40	30
EBA	SHAFT, POWER TAKEOFF (TCI)	10000	200	400	1000

Table 3.10 shows withdrawal and restock times for the parts used in the analysis. The location column indicates whether the part is stocked locally or whether, when needed, it must be acquired from a parts depot. The withdrawal time is the time required to obtain the part from the inventory when the part is in stock. The restock time column shows the normal time required to order and restock a part for the inventory. The emergency restock time is the time required to obtain a part for the inventory when the part is out of stock and is needed immediately. All times are in hours.

Table 3.10. Parts Withdrawal and Restock Times

Fail Mode	Part Description	Location	Withdrawal Time	Restock Time	Emergency Restock Time
EAO	ADG	Depot	168	720	600
EAA	Pump Assy_ADG	Depot	96	332	240
EAB	Pump Assy_JFS	Depot	96	332	240
EAC	Oil Filter	Local	2	332	240
EAD	Oil Delta-P Indicator	Depot	96	332	240
EAE	Torque Converter Assy	Depot	96	720	600
EAF	Clutch Slip Assy	Depot	96	720	600
EAG	Oil Shutoff Valve	Depot	96	332	240
EAH	ADG Clutch Servo Valve	Depot	96	332	240
EAJ	Chip Detector Plug	Local	2	332	240
EAK	JFS Speed Sensor	Depot	96	332	240
EAL	PTO Speed Sensor	Depot	96	332	240
EAM	Solenoid Valve	Depot	96	332	240
EAP	Air Bleeder Check Valve	Depot	96	332	240
EBA	PTO Shaft	Depot	96	332	240

Table 3.11 shows the maintenance and inspection schedule for the ADG. The interval column is the number of flight hours between the scheduled activities. The column labeled "components replaced" shows which components will be inspected and/or replaced by the scheduled activity. Refer to Table 3.10 to identify these components. The cost column includes labor but not the cost of parts, which can be found in Table 3.9. A normal distribution is assumed for the time to perform the maintenance or inspection with the means and standard deviations given in the last two columns of Table 3.11. For the routine and phase inspections, it is assumed that the components are examined and, if it is concluded that the component will likely fail before the next scheduled inspection, the component is replaced. The probability of false positive is the likelihood that the inspection will conclude that a failure is imminent when it is not. Similarly, the probability of a false negative is the probability that the inspection will conclude that a component will perform successfully until after the next inspection when in fact, it will fail.

Table 3.11. Scheduled Maintenance and Inspections

Scheduled Activity	Interval	Components Replaced	Cost	Prob. false positive	Prob. false negative	Mean Time	Std. Dev.
ADG Time Change	2500	EAO	3,000	NA	NA	30	5.00
PTO Time Change	1200	EBA	400	NA	NA	4	0.50
Routine Inspection	60	EAC, EAJ	100	0.02	0.30	1	0.25
Phase Inspection	300	EAC, EAD, EAH, EAJ, EAM, EAP	1,500	0.01	0.10	15	2.50

1.16.2. Sensitivity Analysis

The purpose of the sensitivity analysis is to determine which variables most influence performance of the ADG as measured by its annual maintenance cost, scheduled downtime, and unscheduled downtime. Results of the sensitivity analysis will be used to eliminate variables that have negligible effect on these performance measures in order to simplify the optimization analysis.

1.16.2.1. Variable Definitions and Ranges

The sensitivity analysis evaluated the following variable areas:

- Maintenance and inspection intervals,
- Inspection false-positive probability,
- Inspection false-negative probability,
- Spares restock time,
- Spares withdrawal time, and
- Spares reorder level.

Ranges used in the sensitivity analysis for these variables are presented in the following tables.

Table 3.12 shows the variable ranges used for maintenance and inspection intervals. The sensitivity analysis sampled intervals from a triangular distribution with the nominal interval serving as the most likely value. Note that the routine and phase inspections have been separated by component failure mode. The intervals for the two routine inspections (EAC and EAJ) are highly correlated so that they continue to be inspected together. Similarly the phase inspections are also highly correlated to keep them on the same schedule. However, separating them allows their false positive and false negative probabilities to be evaluated independently (see Tables 3.13 and 3.14).

Table 3.12. Variable Ranges for Scheduled Maintenance and Inspections

Maintenance ID	Nominal Interval	Minimum Interval	Maximum Interval
ADG Time Change	2500	2000	3000
PTO Shaft Time Change	1200	800	1500
Phase Inspection_EAD	300	200	400
Phase Inspection_EAH	300	200	400
Phase Inspection_EAM	300	200	400
Phase Inspection_EAP	300	200	400
Routine Inspection_EAC	60	50	100
Routine Inspection_EAJ	60	50	100

Table 3.13 shows the variable ranges used for inspection false-positive probabilities. The nominal false-positive probability is shown for reference. In each case the false-positive probability is sampled from a triangular distribution with the most likely set to the nominal value.

Table 3.13. Variable Ranges for Inspection False-Positive Probabilities

Maintenance ID	Nominal False Positive	Minimum False Positive	Maximum False Positive
Phase Inspection_EAD	0.01	0	0.2
Phase Inspection_EAH	0.01	0	0.2
Phase Inspection_EAM	0.01	0	0.2
Phase Inspection_EAP	0.01	0	0.2
Routine Inspection_EAC	0.02	0	0.2
Routine Inspection_EAJ	0.02	0	0.2

Variable ranges for inspection false-negative probabilities are given in Table 3.14. For each inspection, the false negative probability is sampled from a triangular distribution with the nominal value as the most likely.

Table 3.14. Variable Ranges for Inspection False-Negative Probabilities

Maintenance ID	Nominal False Negative	Minimum False Negative	Maximum False Negative
Phase Inspection_EAD	0.1	0	0.3
Phase Inspection_EAH	0.1	0	0.3
Phase Inspection_EAM	0.1	0	0.3
Phase Inspection_EAP	0.1	0	0.3
Routine Inspection_EAC	0.1	0	0.3
Routine Inspection_EAJ	0.1	0	0.3

Variables used to evaluate the effects of the spares inventory on ADG performance are in Tables 3.15 through 3.17. Table 3.15 shows the variable ranges used for spare restock times. All times are in hours.

Table 3.15. Variable Ranges for Parts Restock Times

Spare	Used For	Nominal Restock Time	Minimum Restock Time	Maximum Restock Time
ADG	EAO	720	72	1000
Pump Assy_ADG	EAA	332	48	500
Pump Assy_JFS	EAB	332	48	500
Oil Filter	EAC	332	48	500
Oil Delta-P Indicator	EAD	332	48	500
Torque Converter Assy	EAE	720	72	1000
Clutch Slip Assy	EAF	720	72	1000
Oil Shutoff Valve	EAG	332	48	500
ADG Clutch Servo Valve	EAH	332	48	500
Chip Detector Plug	EAJ	332	48	500
JFS Speed Sensor	EAK	332	48	500
PTO Speed Sensor	EAL	332	48	500
Solenoid Valve	EAM	332	48	500
Air Bleeder Check Valve	EAP	332	48	500
PTO Shaft	EBA	332	48	500

Table 3.16 shows variable ranges for spares reorder levels. The reorder level is the stocking level that, when reached, triggers an order for additional parts. In each case, the reorder level is sampled from a triangular distribution with the nominal used as the most likely value. Note that the ADG parts inventory is intended to support 500 aircraft.

Table 3.16. Variable Ranges for Parts Reorder Levels

Spare	Used For	Nominal Reorder Level	Minimum Reorder Level	Maximum Reorder Level
ADG	EAO	6	0	20
Pump Assy_ADG	EAA	1	0	20
Pump Assy_JFS	EAB	1	0	20
Oil Filter	EAC	1	0	20
Oil Delta-P Indicator	EAD	1	0	20
Torque Converter Assy	EAE	2	0	20
Clutch Slip Assy	EAF	1	0	20
Oil Shutoff Valve	EAG	1	0	20
ADG Clutch Servo Valve	EAH	2	0	20
Chip Detector Plug	EAJ	1	0	20
JFS Speed Sensor	EAK	1	0	20
PTO Speed Sensor	EAL	1	0	20
Solenoid Valve	EAM	1	0	20
Air Bleeder Check Valve	EAP	1	0	20
PTO Shaft	EBA	6	0	20

Table 3.17 shows variable ranges for spares withdrawal times. The withdrawal time is the time required to obtain the spare from the inventory to the point where it is needed when the spare is in stock. The times are in hours.

Table 3.17. Variable Ranges for Parts Withdrawal Times

Spare	Used For	Nominal Withdraw Time	Minimum Withdraw Time	Maximum Withdraw Time
ADG	EAO	168	24	336.00
Pump Assy_ADG	EAA	96	24	168.00
Pump Assy_JFS	EAB	96	24	168.00
Oil Filter	EAC	2	0	24.00
Oil Delta-P Indicator	EAD	96	24	168.00
Torque Converter Assy	EAE	96	24	168.00
Clutch Slip Assy	EAF	96	24	168.00
Oil Shutoff Valve	EAG	96	24	168.00
ADG Clutch Servo Valve	EAH	96	24	168.00
Chip Detector Plug	EAJ	2	0	24.00
JFS Speed Sensor	EAK	96	24	168.00
PTO Speed Sensor	EAL	96	24	168.00
Solenoid Valve	EAM	96	24	168.00
Air Bleeder Check Valve	EAP	96	24	168.00
PTO Shaft	EBA	96	24	168.00

1.16.2.2. Sensitivity Analysis Results

The sensitivity analysis was performed using Sandia's SUNS software. A Latin Hypercube sample of size 200 was analyzed. All the input variables were sampled from triangular distributions using the parameters given in the above tables. The outputs of the consequence engine simulation were cost, scheduled downtime, and unscheduled downtime. The cost calculation included the annual cost of inspections, repairs, and parts (purchase, storage/holding cost, and shipping costs) and was normalized to one aircraft. Scheduled downtime includes time spent on scheduled maintenance and inspections while unscheduled downtime results from repairing equipment failures. Both downtime calculations were annualized.

It is important to note that these results should not be interpreted as representing actual variability in the metrics calculated (cost, scheduled and unscheduled downtime). The sensitivity analysis was performed to determine which inputs, if they could be modified, would most influence the output results. The sensitivity analysis will provide a basis for determining which inputs should be considered in the optimization analysis.

Figure 3.9 shows a histogram of annual cost results. The average annual cost is about \$11,500 with a range (5th to 95th percentile) of \$8,750 to \$12,850.

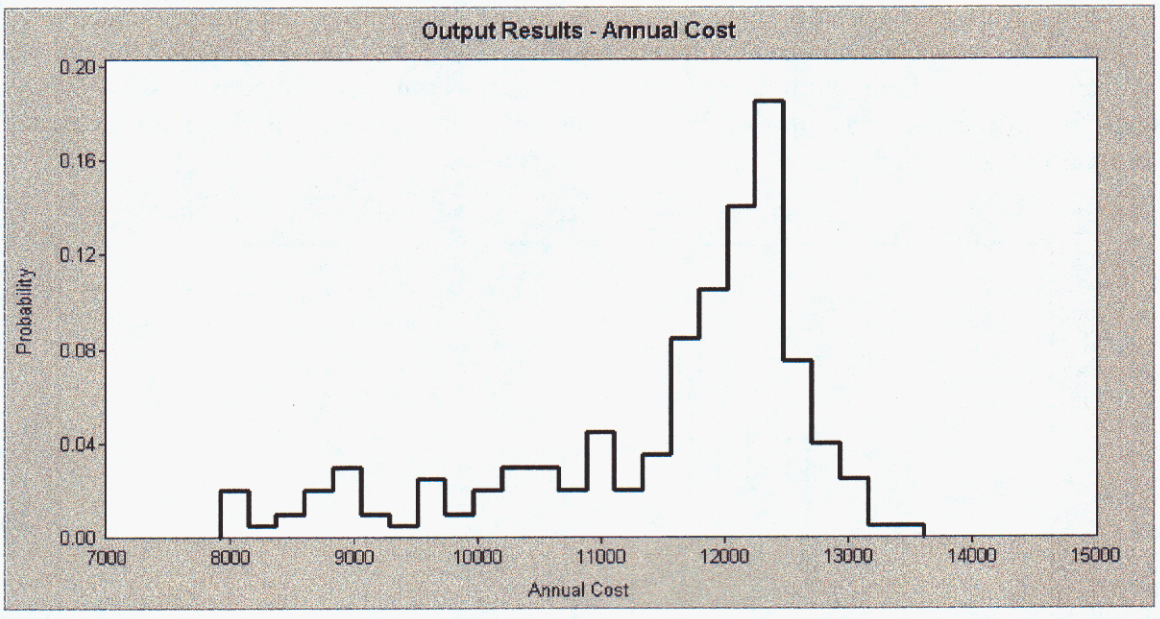


Figure 3.9. Histogram of Annual Cost Results

Figure 3.10 shows the annual scheduled downtime. The average value is about 94 hours with a range (5th to 95th percentile) of 65 to 138 hours.

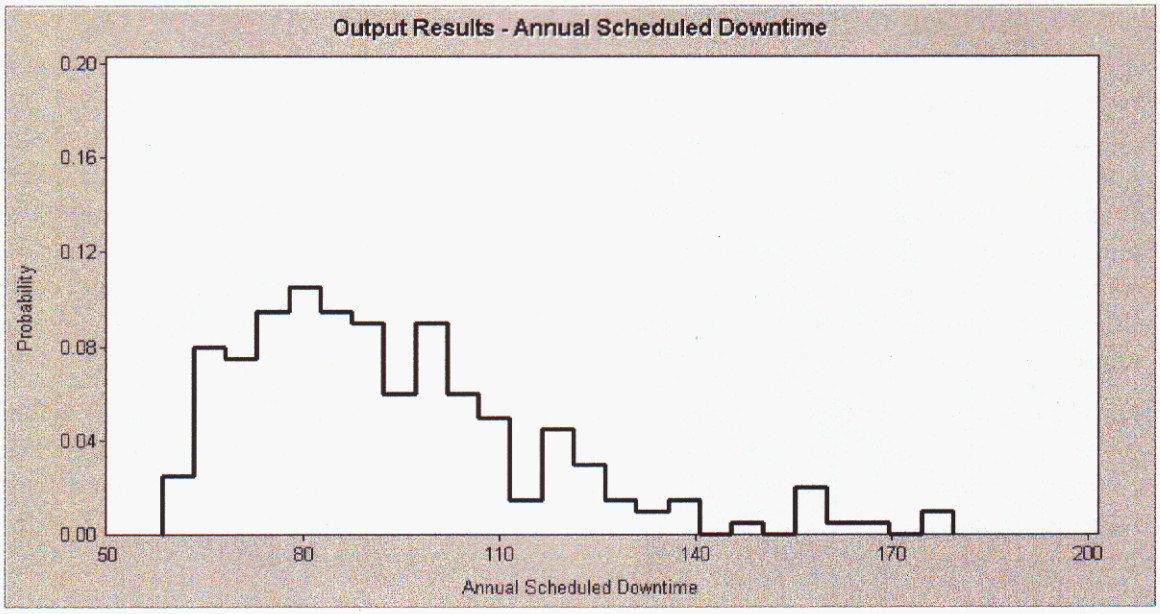


Figure 3.10. Histogram of Annual Scheduled Downtime

Figure 3.11 shows annual unscheduled downtime. The range for unscheduled downtimes is from 43 to 120 hours with an average of 74 hours. Figures 3.10 and 3.11 show clearly that the primary means of avoiding unscheduled downtime is through an extensive program of scheduled inspections and replacements.

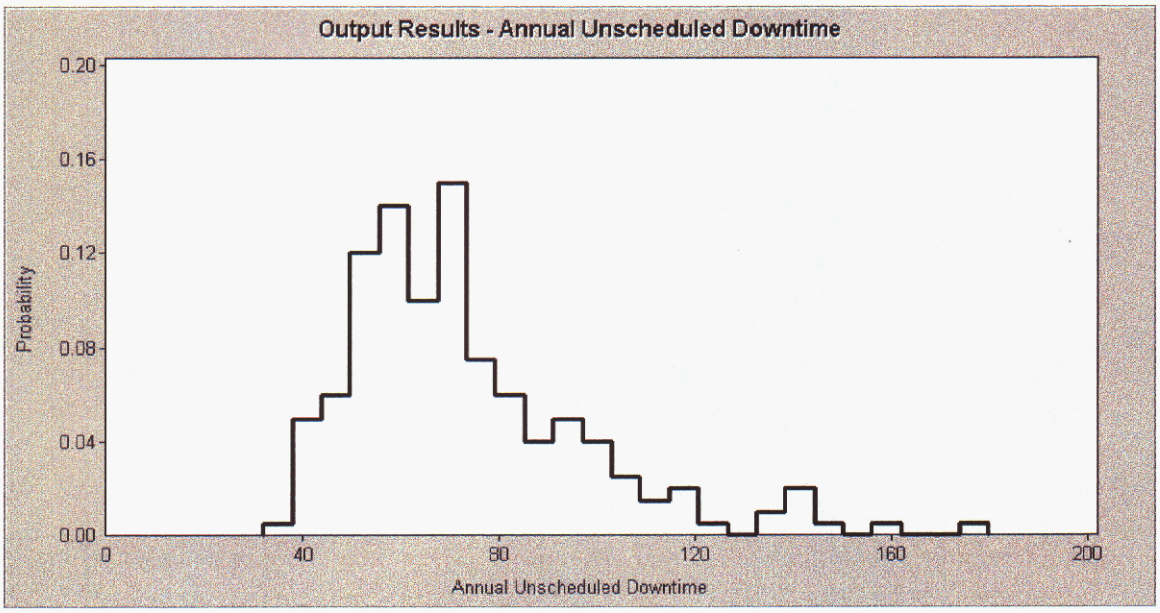


Figure 3.11 Histogram of Annual Unscheduled Downtime

We will use partial rank correlations to identify the input variables that are the largest contributors to the output variables in Figures 3.9, 3.10 and 3.11. A partial correlation measures the strength of a relationship between two variables, while controlling the effect of additional variables. The partial correlation on ranks replaces the raw values of input

and output variables with their ranks. Ranks are assigned by sorting the values in ascending order then assign each value its rank from 1 to N where N is the number of values. In the case of ties, the tied values are assigned their average ranks.

Partial rank correlations for input variables with cost are shown in Figure 3.12. Only the top 9 partial rank correlations are shown.

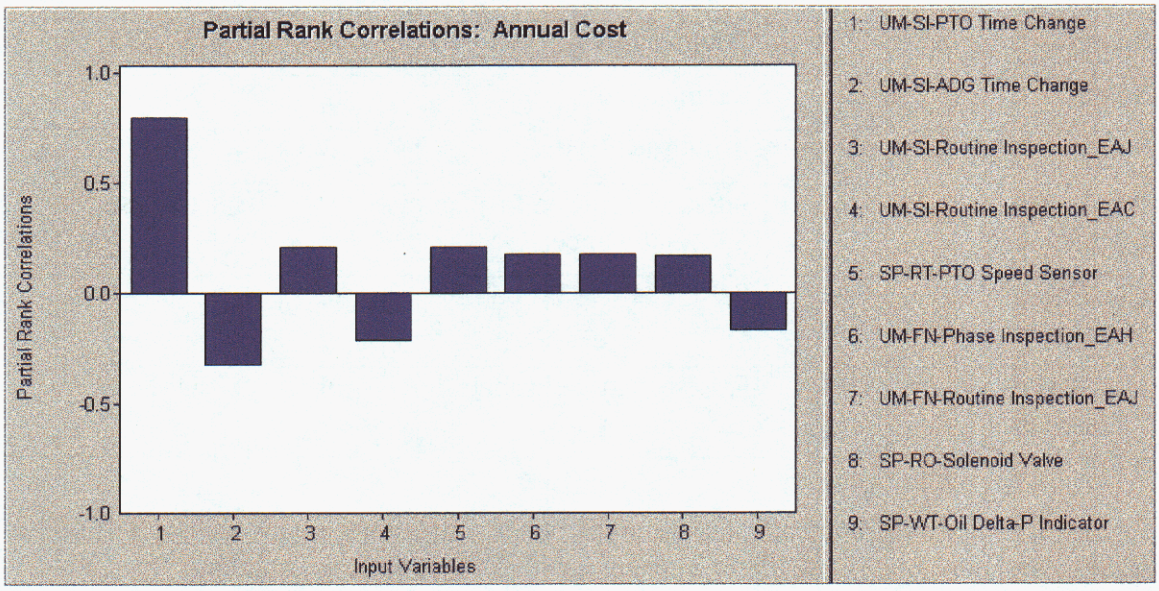


Figure 3.12. Partial Rank Correlations of Input Variables with Annual Cost

A naming convention was applied to the input variables so that the consequence engine could interpret and properly apply the sampled values. In this convention, the first two characters identify the input variable category. For example, SP refers to the spares inventory. The second sequence of two characters identifies the specific variable (e.g., SP-RO refers to the reorder level for a spare). The remaining characters identify the spare or component failure mode to which the variable is applied. For example, SP-RO-PTO Shaft refers to the reorder level for PTO Shaft spares. Table 3.18 shows the naming convention used for the input variables. Use-based maintenance refers to maintenance and inspections that are scheduled based on flight hours.

Using Table 3.18, Figure 3.12 shows that the highest partial rank correlation for annual cost is with the time-change interval for the PTO shaft. The correlation is positive because the sensitivity analysis range (800 to 1500 hours; see Table 3.12) extends to the mean life (1500 hours; see Table 3.6). As the replacement interval for the PTO shaft is increased, the probability increases for a failure to occur before the replacement. The second variable in the Pareto of Figure 3.12 is the time-change interval for the ADG. In this case, the correlation is negative because the sensitivity analysis range does not extend very close to the expected life. As a result, fewer scheduled replacements occur with little increase in the probability of a failure.

The third and fourth highest correlations with annual cost belong to routine inspection intervals for EAJ (Chip Detector Plug) and EAC (Oil Filter). It is interesting that even though these variables are highly correlated, their partial rank correlations with cost are

0.21 and –0.21. Their simple correlations with annual cost are positive and small with the simple raw correlations being about 0.02 and the simple rank correlations being about 0.001.

Table 3.18. Input Variable Naming Convention

Input Category	Code	Input Variable	Code	Full Code
Spares	SP	Reorder Level	RO	SP-RO
Spares	SP	Restock Time	RT	SP-RT
Spares	SP	Withdrawal Time	WT	SP-WT
Use-Based Maintenance	UM	Interval Between Inspections or Maintenance	SI	UM-SI
Use-Based Maintenance	UM	False Positive Probability	FP	UM-FP
Use-Based Maintenance	UM	False Negative Probability	FP	UM-FN

Figure 3.13 shows graphically the strength of the relationship between the PTO Time Change and the annual cost where a strong positive correlation is apparent.

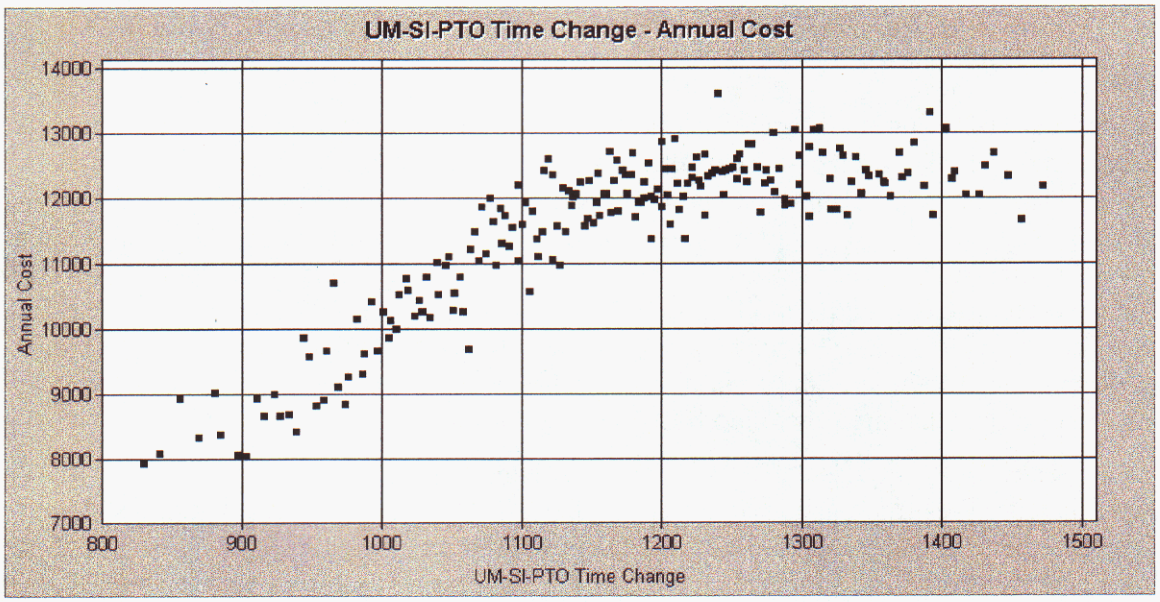


Figure 3.13. Scatter Plot of PTO Time Change Interval vs. Annual Cost

Figure 3.14 shows a Pareto of partial rank correlations between input variables and the annual scheduled downtime. Here the top input variable is the time change interval for the PTO shaft with a partial rank correlation of –0.54. Increasing PTO Shaft time change intervals will necessarily decrease scheduled downtimes. The second and third variables relate to ADG spares with better availability (larger reorder levels) decreasing scheduled downtimes while larger withdrawal times increase scheduled downtimes. It is interesting

that the probability false positives on several inspections are among the top contributors to scheduled downtime because false positives result in unnecessary maintenance.

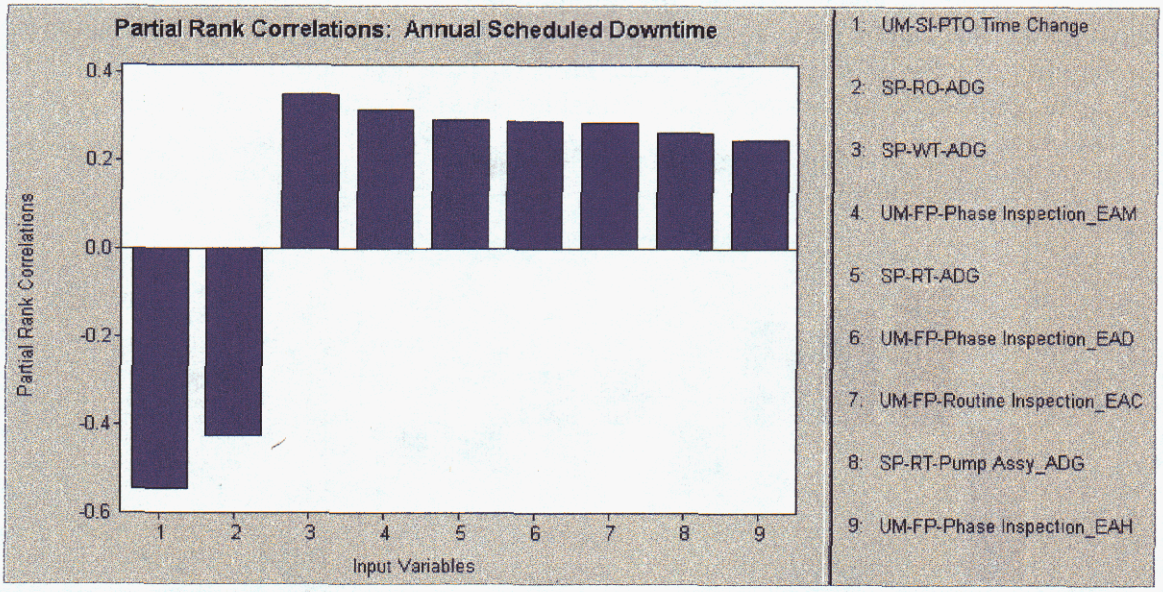


Figure 3.14. Partial Rank Correlations of Input Variables with Annual Scheduled Downtime

Figure 3.15 is a scatter plot of the PTO Shaft time change interval vs. the annual scheduled downtime. While one can see a negative correlation between the variables (partial rank correlation –0.54), the relationship does not appear as strong as that shown in Figure 3.13.

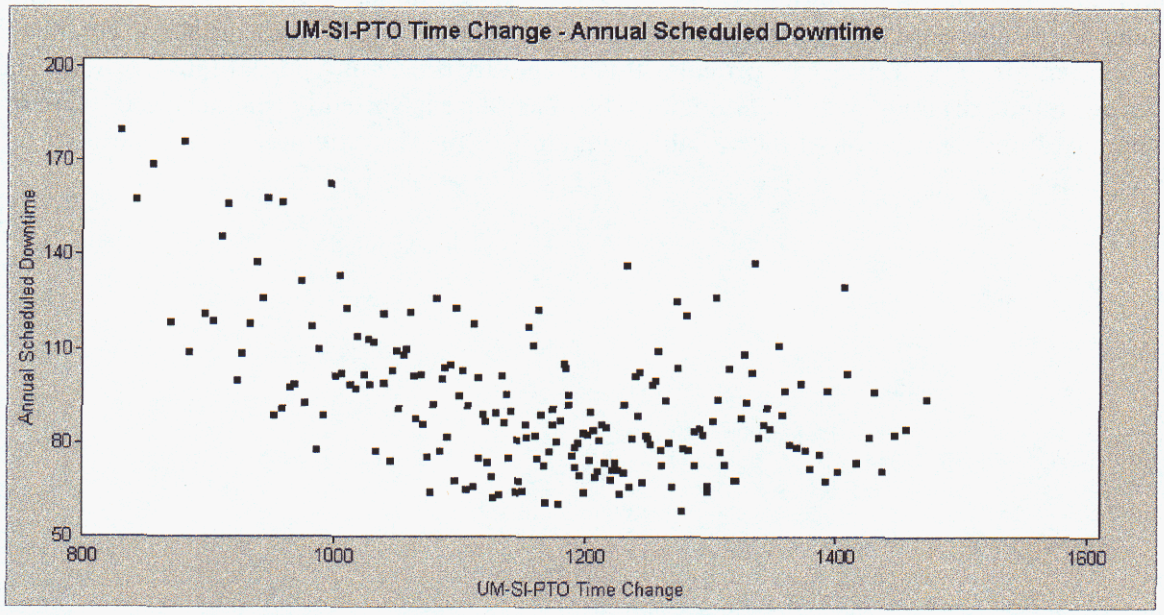


Figure 3.15. Scatter Plot of PTO Shaft Time Change Interval vs. Annual Scheduled Downtime

Figure 3.16 shows partial rank correlations of input variables with annual unscheduled downtime. It is interesting that the top four variables relate to the PTO Shaft. They can be understood from the fact that increasing the time interval for PTO Shaft changes increases the probability of PTO Shaft failures and thereby increases unscheduled downtime. This can be seen from variable 2 in the Pareto of Figure 3.16. With increased probability of PTO Shaft failures, reorder level for parts (variable 1) needs to be higher while the withdrawal and restock times (variables 3 and 4) need to be reduced.

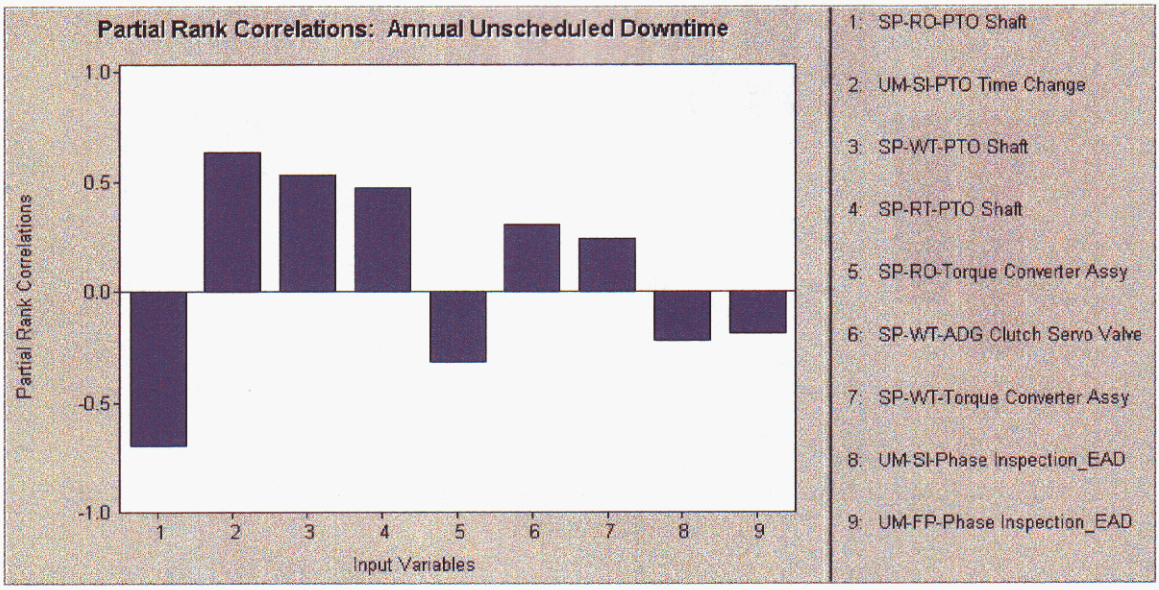


Figure 3.16. Partial Rank Correlations of Input Variables with Annual Scheduled

While the above figures provide interesting insight into the influence of individual input variables on the three output variables (annual cost, annual scheduled downtime, and annual unscheduled downtime), it is difficult to determine which variables are most important overall. Table 3.19 provides this insight by presenting partial rank correlations for all input variables with all three outputs. The table is sorted by the absolute value of the maximum correlation of the variables with any of the three outputs.

Table 3.19. Partial Rank Correlations Sorted by Maximum Absolute Value

Variable	Cost Correlation	Scheduled Downtime Correlation	Unscheduled Downtime Correlation	Maximum Absolute Correlation
UM-SI-PTO Time Change	0.80	-0.54	0.63	0.80
SP-RO-PTO Shaft	-0.03	-0.12	-0.69	0.69
SP-WT-PTO Shaft	-0.04	0.10	0.53	0.53
SP-RT-PTO Shaft	-0.14	0.05	0.47	0.47
SP-RO-ADG	0.00	-0.42	-0.16	0.42
SP-WT-ADG	-0.16	0.35	0.09	0.35
UM-SI-ADG Time Change	-0.32	-0.11	0.04	0.32
SP-RO-Torque Converter Assy	0.09	-0.06	-0.32	0.32
UM-FP-Phase Inspection_EAM	-0.06	0.31	-0.01	0.31
SP-WT-ADG Clutch Servo Valve	0.00	0.16	0.31	0.31
SP-RT-ADG	-0.04	0.29	-0.09	0.29
UM-FP-Phase Inspection_EAD	-0.09	0.28	-0.18	0.28
UM-FP-Routine Inspection_EAC	0.09	0.28	0.00	0.28
SP-RT-Pump Assy_ADG	-0.13	0.26	-0.08	0.26
UM-FP-Phase Inspection_EAH	-0.01	0.24	0.09	0.24
SP-WT-Torque Converter Assy	-0.01	-0.01	0.24	0.24
UM-FP-Phase Inspection_EAP	-0.01	0.24	-0.12	0.24
SP-WT-Solenoid Valve	-0.14	0.22	-0.11	0.22
UM-SI-Phase Inspection_EAD	-0.09	-0.05	-0.22	0.22
UM-SI-Routine Inspection_EAJ	0.21	-0.05	0.06	0.21
UM-SI-Routine Inspection_EAC	-0.21	0.04	-0.06	0.21
SP-RT-PTO Speed Sensor	0.21	-0.09	0.11	0.21
UM-FN-Phase Inspection_EAH	0.18	-0.04	0.00	0.18
UM-FN-Routine Inspection_EAJ	0.18	0.01	-0.09	0.18
UM-FP-Routine Inspection_EAJ	-0.07	0.17	-0.06	0.17
SP-RO-Solenoid Valve	0.17	-0.04	0.01	0.17
SP-RT-Solenoid Valve	0.10	0.00	0.17	0.17
SP-RO-Air Bleeder Check Valve	0.03	-0.17	-0.01	0.17
SP-WT-Oil Delta-P Indicator	-0.17	0.08	0.07	0.17
SP-WT-Pump Assy_JFS	-0.03	0.09	-0.16	0.16
SP-RO-Clutch Slip Assy	0.04	-0.11	-0.16	0.16
SP-RO-ADG Clutch Servo Valve	0.12	-0.15	-0.06	0.15
SP-RO-Oil Filter	0.00	-0.15	-0.04	0.15
SP-RT-Oil Filter	-0.04	-0.03	-0.15	0.15
SP-RT-Pump Assy_JFS	0.06	-0.04	0.15	0.15

Table 3.19 (Cont'd). Partial Rank Correlations Sorted by Maximum Absolute Value

Variable	Cost Correlation	Scheduled Downtime Correlation	Unscheduled Downtime Correlation	Maximum Absolute Correlation
UM-FN-Phase Inspection_EAD	-0.14	0.11	-0.07	0.14
UM-SI-Phase Inspection_EAM	0.07	-0.03	0.14	0.14
SP-RT-Oil Shutoff Valve	-0.14	0.05	0.11	0.14
SP-RO-Oil Delta-P Indicator	0.11	-0.13	-0.05	0.13
SP-RO-JFS Speed Sensor	-0.08	0.02	-0.13	0.13
SP-WT-Oil Filter	0.04	0.13	-0.02	0.13
SP-RT-Clutch Slip Assy	-0.02	0.09	0.12	0.12
SP-RT-Oil Delta-P Indicator	-0.12	0.03	-0.08	0.12
SP-RO-Pump Assy_ADG	0.07	0.05	0.11	0.11
SP-WT-Chip Detector Plug	-0.06	0.11	-0.11	0.11
SP-RT-Torque Converter Assy	0.02	-0.07	0.11	0.11
SP-RT-JFS Speed Sensor	0.11	-0.02	0.01	0.11
SP-WT-PTO Speed Sensor	-0.05	0.11	0.03	0.11
SP-RO-Pump Assy_JFS	-0.10	0.02	-0.11	0.11
SP-RO-Oil Shutoff Valve	-0.06	-0.11	-0.11	0.11
SP-WT-Clutch Slip Assy	-0.09	0.05	0.11	0.11
UM-FN-Phase Inspection_EAM	0.04	0.04	0.11	0.11
UM-FN-Phase Inspection_EAP	0.03	-0.11	0.09	0.11
SP-RT-ADG Clutch Servo Valve	0.07	0.10	0.06	0.10
SP-WT-Air Bleeder Check Valve	0.10	0.07	0.08	0.10
UM-SI-Phase Inspection_EAH	0.09	-0.01	0.03	0.09
SP-RO-PTO Speed Sensor	-0.08	0.04	0.00	0.08
UM-SI-Phase Inspection_EAP	-0.07	0.08	0.04	0.08
SP-WT-Pump Assy_ADG	-0.05	-0.07	-0.03	0.07
SP-RO-Chip Detector Plug	-0.07	-0.04	-0.04	0.07
SP-WT-Oil Shutoff Valve	-0.03	0.00	0.07	0.07
SP-RT-Air Bleeder Check Valve	-0.07	0.03	-0.02	0.07
UM-FN-Routine Inspection_EAC	-0.03	0.05	-0.06	0.06
SP-WT-JFS Speed Sensor	0.04	0.01	0.02	0.04
SP-RT-Chip Detector Plug	0.02	0.02	0.01	0.02

The results in Table 3.19 will be used to reduce the size of the optimization problem by eliminating all variables having maximum partial rank correlation less than 0.22. The scatter plot of the routine inspection interval for the chip detector plug (EAJ) vs. annual cost (Figure 3.17) illustrates the reason for choosing this cutoff. While the partial rank correlation between these two variables is 0.21, it would be hard to discern any relationship by examining the scatter plot. Applying this cutoff will eliminate 46 of the 65 input variables from the optimization analysis. This will reduce the number of possible combinations by a factor of at least 10^{40} and will greatly increase the possibility of find an optimal or near optimal solution.

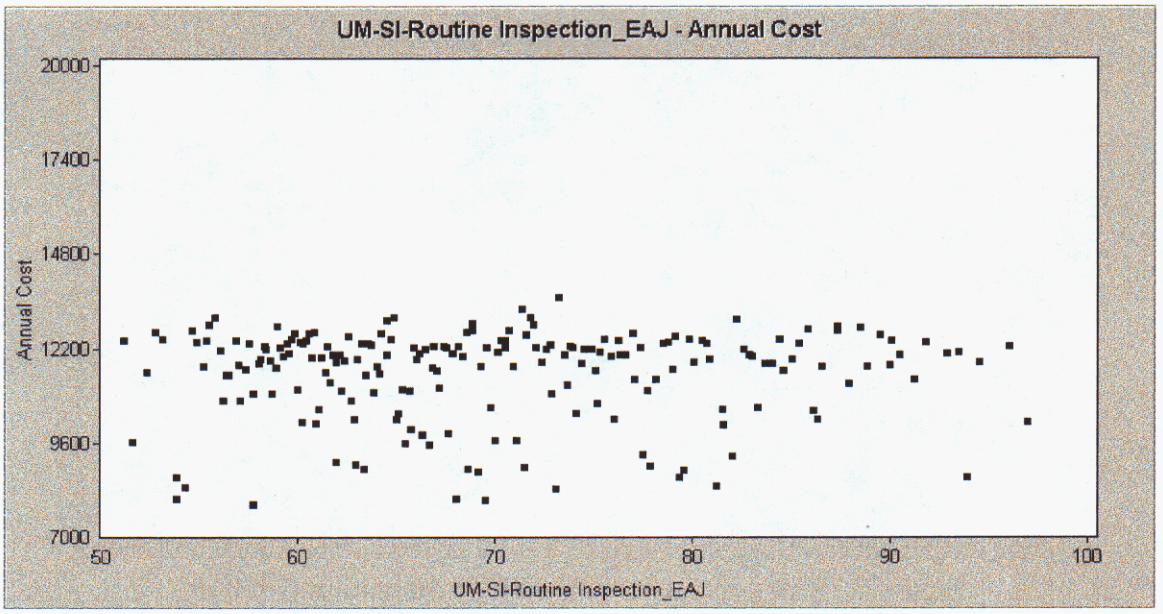


Figure 3.17. Scatter Plot of ADG Pump Assembly Restock Time vs. Annual Unscheduled Downtime

1.16.3. Optimization Analysis

The optimization analysis considered the top 19 variables in Table 3.19. The optimization analysis used Sandia's GO optimization software, which is based on a fairly standard genetic algorithm. Each optimization run used 100 simulations and results were averaged over the 100 simulations. Performance measures optimized were the average annual cost, the average annual scheduled downtime, and the average annual unscheduled downtime.

Because of time constraints, the optimization was only run for 24 generations. The population size was 100 – that is, 100 different input combinations were evaluated on each generation. Input combinations that did not change from one generation to the next were not reevaluated. In the course of the optimization analysis, approximately 1700 different input combinations were evaluated; each of them was run for 100 simulations. The fitness history for the optimization is show in Figure 3.18.

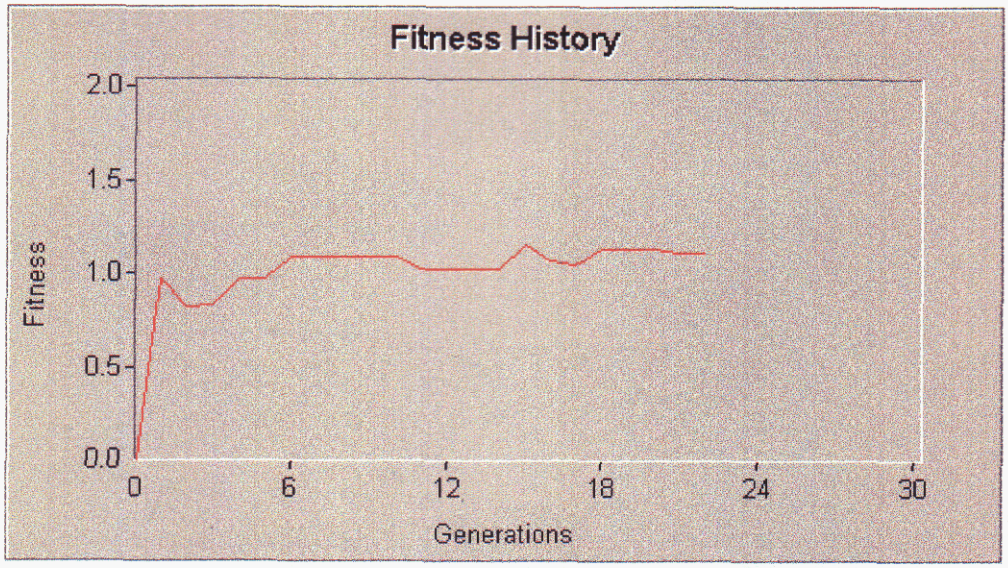


Figure 3.18 Fitness History for the Optimization Analysis

While the optimization could have used more generations to find a better answer, it is clear that progress was made on finding a good result.

Figure 3.19 shows the annual cost history over the 24 generations of the optimization analysis.

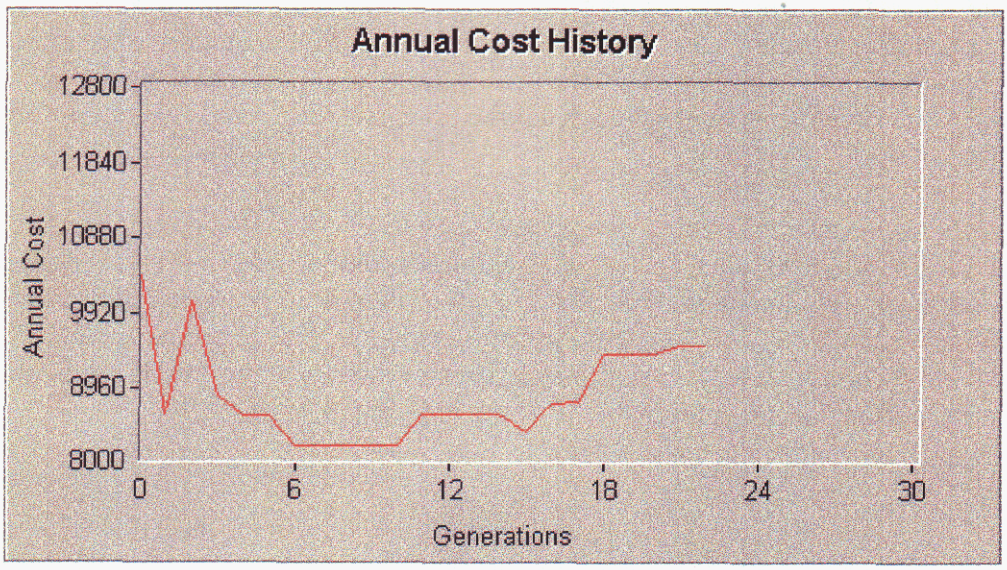


Figure 3.19 Annual Cost History for the Optimization Analysis

Notice that there was an increase in annual cost over the last few generations of the analysis. This is explained by the fact that the annual scheduled downtime decreased considerably at the same time (see Figure 3.20). That is, increased cost was traded for decreased scheduled downtime.

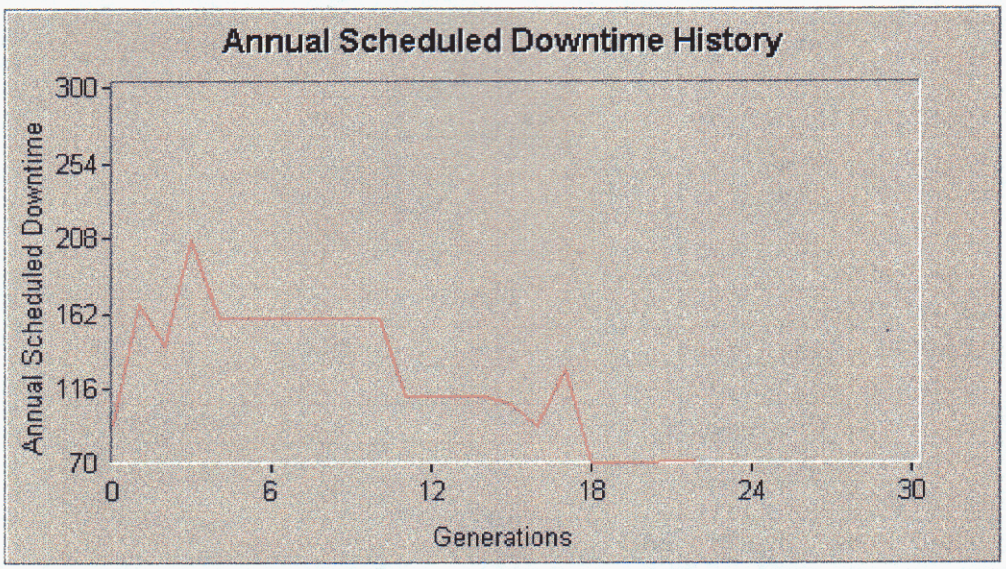


Figure 3.20 Annual Scheduled Downtime History for the Optimization Analysis

Finally, Figure 3.21 shows the history for annual scheduled downtime.

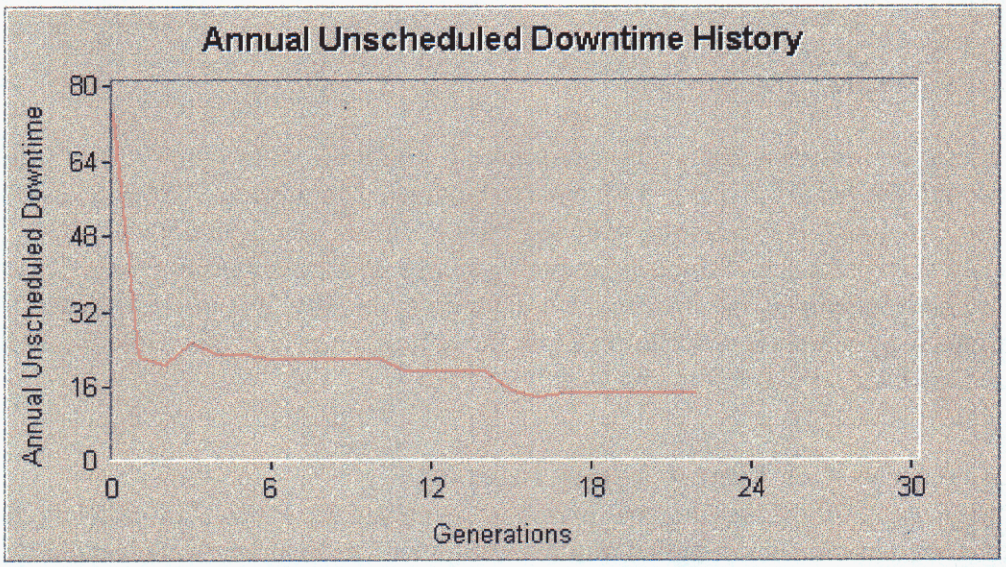


Figure 3.21 Annual Unscheduled Downtime History for the Optimization Analysis

The results of the optimization, in terms of inspection intervals, spares reorder levels, etc. are summarized in Table 3.20.

Table 3.20 Results of Optimization Analysis

Variable	Recommended Value	Baseline Value
UM-SI-PTO Time Change	800	1200
SP-RO-PTO Shaft	12	6
SP-WT-PTO Shaft	24	96
SP-RT-PTO Shaft	240	332
SP-RO-ADG	8	6
SP-WT-ADG	48	168
UM-SI-ADG Time Change	2500	2500
SP-RO-Torque Converter Assy	18	2
UM-FP-Phase Inspection_EAM	0.02	.01
SP-WT-ADG Clutch Servo Valve	24	96
SP-RT-ADG	360	720
UM-FP-Phase Inspection_EAD	0	0.01
UM-FP-Routine Inspection_EAC	0	0.02
SP-RT-Pump Assy_ADG	250	332
UM-FP-Phase Inspection_EAH	0.04	0.01
SP-WT-Torque Converter Assy	24	96
UM-FP-Phase Inspection_EAP	0	0.01
SP-WT-Solenoid Valve	72	96
UM-SI-Phase Inspection_EAD	380	300

Because the optimization analysis needed more generations to develop the best solution that it could, the results in Table 3.20 have to be viewed cautiously. Nevertheless, some results are obvious. For example, reducing false positives on inspections would obviously decrease costs. Similarly, reducing spares withdrawal times would obviously decrease both scheduled and unscheduled downtime. Finally, it is interesting that the PTO time change interval was shortened by the optimization analysis while the ADG time change interval was left unchanged.

1.16.4. Analysis of Time-Change Interval

One of the uses of the Consequence Engine is to analyze the benefits of various types of maintenance actions. For complex military equipment, critical items are usually replaced on a Time-Change Interval (TCI). This means that they are replaced when a certain number of operating hours has been reached: for example, replace an item after it has 1000 operating hours.

To demonstrate the analysis of the Time-Change Interval, we consider an Accessory Drive Gearbox (ADG). For the purposes of this analysis, we have assumed that the ADG is one unit. We did not model components such as gears, bearings, and clutches then roll that data up to an ADG system level, though that might be the next step. We assumed that the ADG had a wearout distribution with 5% probability of burn-in failure during the first 100 hours, 10% probability of failure between burn-in and onset of wearout, and the

end-of-life characterized by a normal distribution with a mean of 4000 hours and a standard deviation of 500 hours.

The Consequence Engine has very detailed logic to simulate preventive maintenance actions and inspections. For preventive maintenance actions, the user can specify whether to repair or replace an item. The user also specifies the "percent renewal" on that item if repaired or replaced. For example, if an item is replaced, its clock is usually set back to zero with 100 percent renewal. However, if an item is repaired, it may not be completely renewed. So, for example, its expected life might be set to 70% of the original life estimate. For this simulation, we assumed that if the ADG failed, it would be replaced with 100% renewal.

For inspections, the simulation looks ahead to the next scheduled inspection and generates a probability that the component will fail before the next inspection. For example, if the inspections are every 300 hours and the clock time on the gearbox is 600 hours, the probability of a failure between 600 and 900 hours is calculated from the TTF distribution, a random number is generated, and if it falls within the probability, a failure event is generated.

If a failure would occur before the next inspection, there is a probability that the current inspection does not detect this impending failure. We take as input a "false negative" probability. For example, if the false negative probability is 30%, and random sampling from the TTF distribution indicates that there would be a failure before the next inspection, there is a 30% chance that the inspection at 600 hours will not detect the pending failure. In that case, the component will not be repaired as a result of the inspection and the component will fail before the next inspection. If the inspection correctly identifies a pending failure (with a 70% chance in this example), then the ADG will be replaced at 600 hours.

If the inspection indicates that the component will not fail before the next scheduled inspection, then we see if a false positive result has occurred, meaning that the inspection indicated a pending failure when there was none, and the component was replaced prematurely. We use a 1% false positive rate in this analysis.

We use the inspection feature in the Consequence Engine combined with the false positive/false negative rates to model a PHM system. Note that we are not modeling a continuous PHM system, since we are only checking the inspection results every 300 flight hours, but we could reduce the inspection interval to say 50 or 100 hours to represent a prognostic system operating continually.

The results of this analysis are shown in Figure 3.22. This graph has the probability of a false negative on the X-axis. Larger numbers mean that the inspection (i.e. prognostics) is not that good at detecting onset of failure. On the Y-axis, we have the probability of ADG failure per year, per aircraft. The different lines show the results of the analysis, parametrically varying the time-change interval Tc from 2500 hours to 4000 hours.

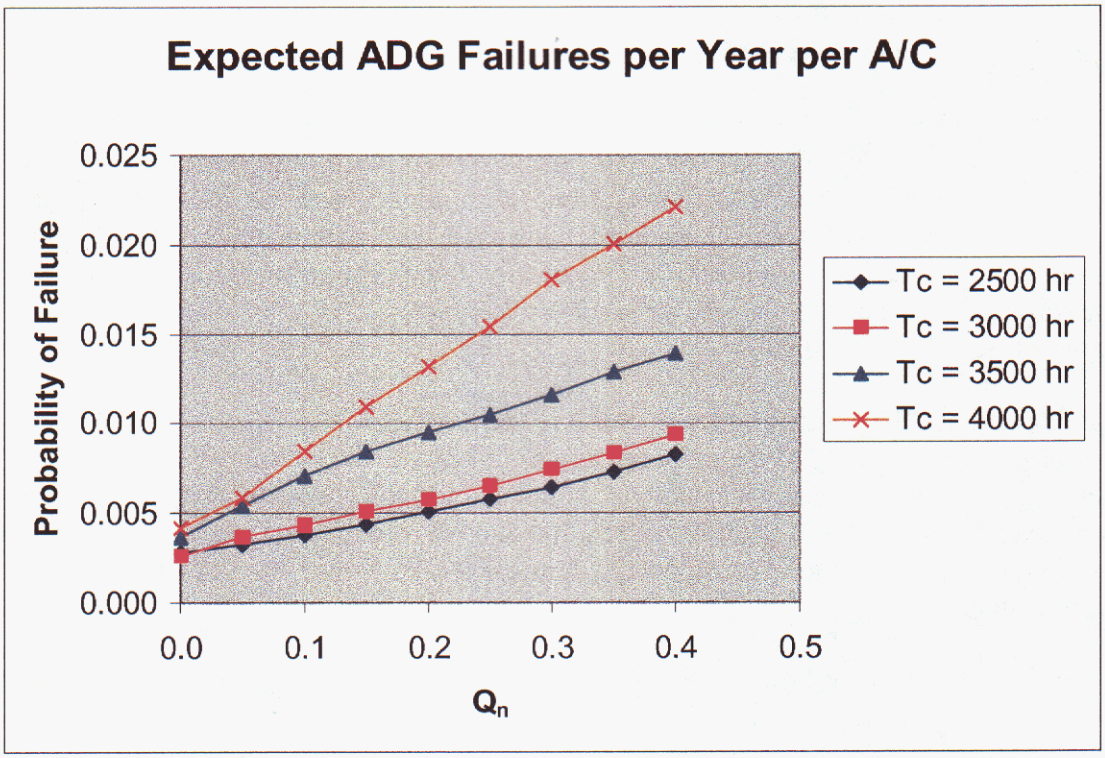


Figure 3.22. Annual ADG Failures per A/C per Year

Look at the lowest line first. This shows that with the current replacement interval of about 2500 hours, there is approximately a 0.3% chance of failure per aircraft per year, assuming a perfect inspection (this is the baseline failure rate primarily due to burn-in failures that occur before an inspection is scheduled.) Even with a false negative rate of 40%, the probability of ADG failure is still under 1% because the time change is at the 3σ limit on the wearout portion of the assumed time-to-failure distribution (4000-3*500). As the time change interval is lengthened, the probability of ADG failure increases. However, the assumption that many of the failures are caught at the 300 hour inspections means that overall, the ADG failure rate does not increase significantly. In the worst case scenario, with a 40% false negative rate and a time change at 4000 hours, the failure probability is around 2.2% per aircraft per year.

This graph can be used to determine the required precision of a prognostic system to maintain a certain reliability level. If, for example, an annual failure probability for the ADG per aircraft was desired to be 0.5% or less, the accuracy of a prognostic system with a 3500 hour time change interval would require a false negative probability of around 3%. If an annual failure probability for the ADG per aircraft was desired to be 1% or less, the accuracy of a prognostic system with a 4000 hour time change interval would require a false negative probability of approximately 13%.

We have performed some validation on this analysis, and run the simulation long enough to get good statistics. Each dot in Figure 3.22 represents five million flight hours, or approximately 16,667 aircraft years. We have analytically calculated the number of

failures expected during burn-in and the number of failures that will not be caught by the inspection due to the false negatives, and these numbers come very close to the failure rates shown in Figure 3.22. Note that if there were no inspections and no time changes, the possible number of failures could be as high as 7.5%/year, assuming the 4000 hour mean life. The field failure data indicated that there were 81 remove and replace actions on the ADG that were not time changes over a two year period, with data from 1184 aircraft. That translates to 40/year or about 3.5% of the fleet per year. It is difficult to say, however, how many of these remove and replace were true failures – many of them may have been incidents where maintenance personnel thought something was not quite right and sent the ADG for replacement (or the ADG could have been removed and replaced though another component such as the JFS was the culprit). It is also not clear at this point how good the current inspection at the 300 hour phase is at predicting failure over the next 300 hours, given that maintenance personnel currently can only check the magnetic chip detector and the oil filter/oil level. The current inspection probably has some PHM capability, but does it predict more than 50% of impending failures? We need to collect more data to determine that. The point to remember is that the results in Figure 3.22 do assume that many of the failures are caught by the 300 hour inspection, so the overall probability of failure numbers are lower than would be in absence of a PHM capability.

Summary

Given the state-of-the-art in prognostics, we suggest the following as a roadmap for developing a PHM system for a particular component or system:

1. Create a reliability model of the system.
2. Determine the critical components and failure modes, in terms of cost, frequency, and severity.
3. For each critical component, identify what factors could help update the remaining useful lifetime estimate. This may include:
   - Sensor data
   - Operational usage such as flight parameters, environmental parameters
   - Previous maintenance/inspection data
   - Component age
4. For each component, determine how these factors (sensor data, operational usage, maintenance records, etc.) will be used to estimate the remaining useful life. Specifically, identify potential algorithms given an understanding of the data streams. This is the hard part, the "science" of prognostics. It may involve significant experimental setups to test under a variety of regimes so that enough data is available to develop prognostic algorithms with some factor of confidence.
5. Develop methods for analyzing the consequence of potential part failures and how to schedule maintenance that will prevent this.
6. Determine how PHM will integrate with the logistics and maintenance program.

The concept of machine prognostics as a means of increasing reliability and decreasing maintenance costs has spurred considerable research and development into prognostics but much remains to be done to have a highly accurate, operational system. A very important factor for ensuring success of PHM on a particular system will be to take a systems analysis approach and simulate the likely behavior of various prognostics approaches. This will be critical to understand how uncertainty in failure rates, in bad "threshold" values, in order and repair times, and most importantly in the prognostic accuracy will affect the performance of PHM. Proper sensitivity analyses of reliability models and maintenance/restocking models will help one understand which prognostics methods will be cost effective and what cost improvements one can realistically expect (trade studies).

This report has summarized the results of PHM algorithms and software tools developed as part of an LDRD project. The Evidence Engine integrates information from a variety of sources in order to take into account all the evidence that impacts a prognosis for system health. The Evidence Engine has the capability for feature extraction, trend detection, information fusion through Bayesian Belief Networks (BBN), and estimation of remaining useful life. The Consequence Engine involves algorithms to analyze the consequences of various maintenance actions. The Consequence Engine takes as input a maintenance and use schedule, spares information, and time-to-failure data on components, then generates maintenance and failure events, and evaluates performance measures such as equipment availability, mission capable rate, time to failure, and cost. The capabilities represented in both the software tools and algorithms described in this report provide a foundation for prognostic system development at Sandia in the future.

References

1. Banks, Jeffrey and Ken Maynard. *Prognostics and Health Management.* A research report compiled by Pennsylvania State University's Applied Research Laboratory, March 2001.
2. Berry, James E. *Illustrated Vibration Diagnostic Wall Chart, 4th Edition.* Technical Associates of Charlotte, P.C. (available at www.technicalassociates.net/wallchart.html).
3. Christodoulou, Leo. *Prognosis*. Overview talk for DARPA Broad Agency Announcement Call 03-02, Bidders Conference and Workshop, Sept. 27, 2002, Alexandria, VA.
4. Engle, Stephen J., Barbara J. Gilmartin, Kenneth Bongort, and Andrew Hess. *"Prognostics, the Real Issues involved with Predicting Remaining Useful Life."* Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 2000.
5. Golay, M. W. and C. W. Kang, "*On-Line Monitoring for Improved Nuclear Power Plant Availability and Operational Advice -Active Equipment Monitoring: Rotating Machinery*," pp 12-13, MIT-ANP-TR-057, Vol. I, Massachusetts Institute of Technology, February, 1998.
6. Hess, Andrew J. and Bill Hardman. "SH-60 Helicopter Integrated Diagnostic System (HIDS) Program Experience and Results of Seeded Fault Testing." Proceedings of the Workshop on Helicopter Health and Usage Monitoring Systems, DSTO-GD-0197. Melbourne, Australia, Feb. 1999.
7. Idaho National Engineering Laboratory, *NRC Plant Performance Database*, 1996.
8. Jensen, Finn V. *Bayesian Networks and Decision Graphs*, Statistics for Engineering and Information Science Series, Springer-Verlag, 2001.
9. Kacprzynski, Gregory J., Michael J. Roemer, Girish Modgil, Andrea Palladino, and Kenneth Maynard, *"Enhancement of Physics of Failure Prognostic Models with System Level Features."* Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 2002.
10. Kacprzynski, Gregory J., Micheal J. Roemer, Andrew J. Hess. *"Health Management System Design: Development, Simulation, and Cost/Benefit Optimization."* Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 2002.
11. Kohonen, Teuvo. *Self-Organizing Maps, 3rd edition*, Springer-Verlag, 2001.
12. Nuclear Energy Research Institute (NERI) Report: SMART-NPP-I-7-02. Smart Nuclear Power Plant, Year 3 Annual Report, 2002.
13. Roemer, Michael J., Gregory J. Kacprzynski, and Rolf F. Orsash. *"Assessment of Data and Knowledge Fusion Strategies for Prognostics and Health Management"*, Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 2001.
14. Society for Machinery Failure Prevention Technology website, Gear Diagnostic Parameters: www.mfpt.org/mfpt_geardiagnosticparam.html.
15. Society for Machinery Failure Prevention Technology, *Impact of Prognostics on Organizational Success*. Proceedings of the 57th MFPT Meeting. Virginia Beach, VA, April 14-18, 2003.
16. Tobias, Paul A. and David C. Trindade, *Applied Reliability, 2nd Edition*, Van Nostrand Reinhold, 1995.

17. Vachtsevanos, George. *Fault Diagnostics/Prognostics for Equipment Reliability and Health Maintenance.* Course notes from a Georgia Institute of Technology Continuing Education Course, Atlanta GA, May 20-23, 2002.
18. Wowk, Victor. *Machinery Vibration: Measurement and Analysis.* McGraw Hill, 1991.

Appendix A: Software Objects in Evidence Engine

BBN class: The BBN class is an object shell for containing a Hugin Bayesian belief network. The class allows one to load a BBN model, compile it, identify chance, decision, or utility nodes, modify the values of these nodes, and propagate updated values through the BBN. The BBN class does not allow the user to modify the topology of the BBN model: that is taken as fixed from the BBN model file.

- Chance node: contains a number of properties relating to the node, such as node index, node label, number of states of the node, node "belief" value, and "finding" value (evidence).

- Decision Node: contains a number of properties relating to the node, such as node index, node label, number of states of the node, node value (expected utility).

- Utility Node: contains a number of properties relating to the node, such as node index, node label, number of states of the node, node value (expected utility).

Table A.1 Properties of the BBN Class

Property	Type	Description
BBNFileName	String	Return the BBN File Name
ChanceNodes	ChanceNodes	Define the chance nodes for the BBN
DecisionNodes	DecisionNodes	Define the decision nodes for the BBN
UtilityNodes	UtilityNodes	Define the utility nodes for the BBN

Table A.2 Methods of the BBN class

Method	Description
LoadNetwork	Loads a BBN network in the HUGIN format, returns an integer specifying file open success or problem
Compile	Compiles a BBN network
Propagate	Propagates evidence (changed node values) through the BBN Network
SaveNetwork	Save the network to the BBNFileName

Distribution class: The distribution class is quite large, with a number of properties and methods for calculating cumulative distribution values for a variety of distribution types. The distribution class has an enumerated type of distributions, consisting of the following: Fixed, Wearout, Exponential, Uniform, Triangular, Normal, and Weibull. The class has an indexed property, Pars(), which is used to store the parameters of the various distributions. The class has a function DistVal which returns a value from the specified distribution. If a particular cumulative probability value is not input, DistVal returns a randomly sampled value. Otherwise, it returns the value corresponding to the input probability. If a minimum x is included, the value returned is conditioned on x >

MinX. Another important function is ProbVal, which returns the cumulative probability value from the distribution for a particular X value. ProbVal also can be conditioned on x>MinX.

Methods for calculating the normal, uniform, fixed, weibull, triangular, and exponential distribution are coded into this class. The wearout distribution is a special case since it is really a piecewise continuous set of three distributions: a burn in period, an constant failure rate period, and a normal distribution around the end of life. Hazard rate(s) and TempWearout are classes that support the calculation of the wearout distribution values.

- Hazard Rate
- Hazard Rates (a collection class)
- TempWearout

Table A.3 Properties of the Distribution Class

Property	Type	Description
DistType	Integer	An enumerated value that specifies the type of distribution (exponential, weibull, etc.).
Pars( )	Single	An indexed property that contains distribution parameters (e.g., shape, location, and scale parameters for a weibull)

Table A.4 Methods of the Distribution class

Method	Description
Clone	Create an exact copy of the distribution object.
DistVal	Returns a randomly sampled value of the dependent variable conditioned on the variable being greater than a specified value.
DistProb	Returns the probability associated with a value of the independent variable conditioned on the variable being greater than a specified value.

Sensor class: The sensors class is a collection class. The sensor class allows one to read in batch or continuous data. The data must be in the following format:

Continuous data: The first line contains the name of the sensor in quotation marks, followed by the number of samples in the data set. Subsequent lines contain the actual data, each line containing a date, value pair. The dates are given in the format of double precision numbers on these lines.

"Accelerometer1",1000
36707.5,1.09273267388344
36707.5416666667,.865884444713592
36707.5833333333,1.08909636855125
36707.625,.882708888053894
36707.6666666667,1.0783764564991
36707.7083333333,1.00310837030411

```
36707.75,1.10995877146721
36707.7916666667,1.13751914024353
36707.8333333333,.976753317117691
36707.875,1.23455493688583
36707.9166666667,1.05088762640953
36707.9583333333,1.02392691135406
36708,1.10494699597359
```

Batch data: The first line contains the name of the sensor in quotation marks, followed by a comma and the number of batches. Each batch must be started by the line "Batch #". The line following the batch number has the date (written in date format), followed by the number of data points in that batch. The succeeding lines are followed by lines with the actual data, each line containing a date, value pair. The dates are given in the format of double precision numbers on these lines. When the batch is complete, a line with "" is written. See the example below, where there are 20 batches of viscosity data, each with 10 values per batch:

```
"VISCOSITY",20
"Batch 1"
#2001-01-01 01:00:00#,10
36892.0416666667,1.14692735075951
36892.0833333333,.927204222679138
36892.125,.950925608873367
36892.1666666667,.987355444431305
36892.2083333333,.931533955335617
36892.25,1.08984436988831
36892.2916666667,1.16862528443336
36892.3333333333,1.07239606618881
36892.375,1.0169931447506
36892.4166666667,1.16672363758087
""
"Batch 2"
#2001-01-01 11:00:00#,10
36892.4583333333,1.27085157838963
36892.5,1.17596231007654
36892.5416666667,1.34218597678053
36892.5833333333,1.32196277753258
36892.625,1.20795990548392
36892.6666666667,1.32205858898001
36892.7083333333,1.34908635850683
36892.75,1.33319824986211
36892.7916666667,1.39589663496587
36892.8333333333,1.46082323394113
""
```

Table A.5 Properties of the Sensor Class

Property	Type	Description
DataType	Enumerated	An enumerated type specifying batch (0) or continuous (1) data
NumBatches	Long	Number of batches
NumDataValues	Long	Number of data values
DateValues()	Double	Array of date values (holds one batch)
DataValues()	Single	Array of sensor data values (holds one batch)
BatchDates	Date	Array of batch dates
FileName	String	File Name containing sensor data

Table A.6 Methods of the Sensor class

Method	Description
ReadFile	Reads data from the file named FileName – forks to one of the two methods below, depending on data type.
ReadBatchData	Reads batch data
ReadContinuousData	Reads continuous data

Sensor Feature Class: The SensorFeatures class is a collection class. The Sensor Feature class allows one to process sensor data, extract sensor features from the input sensor data, and extrapolate or project the sensor feature history to a threshold value. The Sensor Feature class holds both sensor input as well as the feature history, though if the sensor input is in batch form, the sensor feature class only holds the current batch of sensor data since it only processes the feature on one batch at a time. There are many feature extraction methods in this class and we expect to develop more complex feature extraction methods. The current extraction methods are: mean, higher moments (variance, skew, and kurtosis), and RMS. Each extraction method extracts one feature value per sensor batch. For example, if the extraction method is variance and there are 20 batches of data, the feature history will hold 20 variance values.

Once the feature history is populated by an extraction method or by directly reading in sensor feature values, extrapolation of that sensor feature to a threshold may be performed. Currently the extrapolation methods are linear or polynomial regression: one can specify the order of the polynomial in the LeastSquaresProj method.

Table A.7 Properties of the Sensor Feature Class

Property	Type	Description
NumHistory	Long	The number of sensor feature values
NumPoints	Long	The number of points in the raw sensor data array
FeatureDate()	Date	The date associated with the current feature value (indexed from 1 to NumHistory)
FeatureHistory()	Double	The sensor feature value associated with the current feature

		(indexed from 1 to NumHistory)
SensorInput()	Double	Value of the sensor data (indexed from 1 to NumPoints)
ID	String	String that uniquely identifies the sensor feature

Table A.8 Methods of the Sensor Feature Class

Method	Description
Clone	Create an exact copy of the current model object.
ReadFeatureData	Read feature data (data that has already been processed, so extraction does not need to be performed)
TimeToThreshold	Calculates the time at which the current sensor feature history array will cross the critical threshold
RMS	Calculates the Root Mean Square for a set of sensor data
Mean	Calculates the mean for a set of sensor data
HigherMoments	Calculates the variance, skew, and kurtosis for a set of sensor data
LeastSquaresProj	Calculates the least squares regression for the data in the Feature History array. The number of coefficients desired in the regression is input, and the coefficient values are output.

Failure Mode Class: The FailureModes class is a collection class. The failure mode class contains information about a failure mode: its ID, Time-to-Failure distribution, the state the failure mode is in (operational, under repair, etc.) The PHM prototype does not use all of the properties of the failure mode class – primarily we use the SensFeatures property to identify the sensor features associated with the failure mode, the TTF distribution, the age of the failure mode, and the update method (is the failure mode health updated via sensor data, a BBN, or age alone?)

Table A.9 Properties of the Failure Mode Class

Property	Type	Description
ID	String	A unique string that identifies a failure mode.
Description	String	A brief description of the failure mode.
TotTime	Single	The total time in hours since this failure mode was repaired or replaced.
RelUtil	Single	The utilization of this component or failure mode relative to the equipment utilization
Cost	Single	The cost of the equipment failing by this failure mode.
TTFDistribution	Distribution	The time-to-failure distribution for this failure mode. The Distribution object is described below.
FailProb	Single	This is a read-only property giving the probability this failure mode will occur in a specified time interval.
FailTime	Single	A read-only property giving a randomly sampled time until the next occurrence of this failure mode.
NominalDT	Single	The nominal time the equipment is off line when this failure mode occurs.

DTDistribution	Distribution	The down time distribution for this failure mode. The Distribution object is described below.
RepairProb	Single	This is a read-only property giving the probability this failure mode will be repaired in a specified time interval.
RepairTime	Single	This is a read-only property giving a randomly sampled time at which repair will be complete.
Age	Single	The age of the failure mode (usage or operational time)
UpdateType	Enumerated	The method by which the failure mode age is updated: via sensor data, a BBN, or age alone.
SensFeatures	SensorFeatures	A collection of sensor features relating to this failure mode

Combined Distribution Class: The combined distribution class allows one to input three failure mode state probabilities and associated TTF distributions, and updates the overall TTF distribution for the failure mode based on an approach explained in the next section. The inputs to this class are the TTFs of the three states, the probabilities of the three states at a particular time, and the conditioning time for each state. The result is a combined TTF distribution (CDF value) and the inverse of the combined CDF value. Because it bears the same properties and functions, the Combined Distribution class is completely interchangeable with the existing Distribution class described above.

One must assign 3 input distributions and 3 state probabilities to create a well-defined Combined Distribution object. The state probabilities should sum to unity or a warning message is generated (calculation is allowed to proceed). One has the flexibility to independently condition each input distribution with the ElapsedTime properties. The x-value reported in the DistVal function behaves the same way with respect to conditioning as in the Distribution class. However, if the input distributions are conditioned, those x-values are not added to the output since they might not all be the same. There may be valid reasons to have different conditioning times for each input distribution.

Note that state probabilities are not explicitly named severe, intermediate or normal. They are simply indexed. It is up to the user of the class to be consistent with the indices for the input distributions and their corresponding state probabilities.

Table A.10 Properties of the CombinedDistribution Class

Property	Type	Description
DistType	Integer	An enumerated value that specifies the type of distribution (exponential, weibull, etc.).
Pars( )	Single	An indexed property that contains distribution parameters (e.g., shape, location, and scale parameters for a weibull)
Probability of being in States 1, 2, 3		
Conditioning Time for States 1, 2, 3 CDF		

Conditioning Time for Combined CDF		

Table A.11 Methods of the CombinedDistribution class

Method	Description
Clone	Create an exact copy of the distribution object.
DistVal	Returns a randomly sampled value of the combined distribution variable conditioned on the variable being greater than a specified value. (Inverse Combined CDF Value)
DistProb	Returns the probability associated with a value of the combined distribution variable conditioned on the variable being greater than a specified value. (Combined CDF Value)

Appendix B: Enumerations used in the Consequence Engine

B.1 Equipment States

Possible system states are enumerated as follows:

Operational = 1
Operational1 = 2
Operational2 = 4
Other Up Time = 8
Scheduled Down = 16
Unscheduled Down = 32
Shut Down = 64
Nonscheduled Time = 128

Operational
This is the normal operational state for the equipment.
Operational1
This is an additional operational state characterized by accelerated (decelerated) aging of specified failure modes and increased (decreased) performance metrics. This state must be scheduled and acceleration factors and performance-metric-multipliers can be specified for each scheduled occurrence (not yet implemented).
Operational2
This is an additional operational state characterized by accelerated (decelerated) aging of specified failure modes and increased (decreased) performance metrics. This state cannot be scheduled but can be transitioned to. For example, a specific failure mode might leave the equipment in a partially up state. The acceleration factors and performance-metric-multipliers can be specified only once (not yet implemented).
Other Up Time
This state is for times when the equipment is operating but not performing its intended function. An example might be equipment being tested. This is a scheduled state.

Scheduled Downtime
This is the equipment state when maintenance is scheduled.

Unscheduled Downtime
This is the equipment state when it has failed. This state can be transitioned to when a failure mode occurs.

Shut Down
Time when the equipment is powered down without having failed. This is a scheduled state.

Non-Scheduled Time
This is the equipment state when it is not scheduled for use.

B.2 Event Types

The enumerated values for event types are as follows:

Failure Mode Occurs = 0
Preventive Maintenance = 1
Simulation = 2
Workday Segment = 3
Special Period = 4
Other Scheduled = 5

B.3 Time Units

The enumerated values for time units are as follows:

Day = 0
Week = 1
Month = 2
Year = 3

B.4 Failure Mode States

Allowed failure mode states are enumerated as follows:

Up = 1
Incipient = 2
Failed = 3
Scheduled Down = 4

B.5 Preventive Maintenance Action

Enumerated values for preventive maintenance action are:

Repair = 0
Replace = 1

B.6 Preventive Maintenance Decision

Enumerated values for preventive maintenance decisions are:

None = 0
Inspect = 1
Test = 2

Appendix C: Distributions available in the Consequence Engine

The following distribution types are available in the Consequence Engine.

Fixed = 0
Wear-Out = 1
Exponential = 2
Uniform = 3
Triangular = 4
Normal = 5
Weibull = 6

These distributions and their parameters are described in the following sections. When entering parameter values, they must be entered in the order shown here.

C.1 Fixed

The fixed distribution returns the same value every time it is sampled.

Value
This is the single value to be returned when the distribution is sampled. Its units are hours.

C.2 Wear-Out

The "Wear-Out" is a three-part distribution developed for use as a time-to-failure distribution. Its three parts are burn-in, normal life, and wear out. During the burn-in period, the failure rate is assumed to be linearly decreasing. During the normal life, a constant failure rate is assumed. In the wear out portion of the distribution, the time-to-failure distribution is assumed to be normal. The wear-out distribution requires five parameters as follows:

Burn-In Fraction
This parameter determines the fraction of failures that occur during the burn-in period.

Burn-In Duration
This parameter sets the duration of the burn-in period. Its units are hours.

Random Fraction
This parameter sets the fraction of failures that are assumed to occur during the component's normal life.

Mean
This is the mean of the normally-distributed wear-out portion of the distribution. Its units are hours.

Standard Deviation
This is the standard deviation in hours of the normally-distributed wear-out portion of the distribution.

C.3 Exponential

Using an exponential distribution for time-to-failure is equivalent to assuming a constant failure rate.

Failure Rate

This single parameter specifies the constant failure rate (failures per hour) that characterizes the exponential distribution when it is uses for time-to-failure. If you use the exponential distribution as a time-to-repair distribution, the parameter is repair rate (repairs per hour).

C.4 Uniform

The uniform distribution treats every value within the specified range as equally likely.

Minimum

The lower bound of the range of values to be sampled (hours).

Maximum

The upper bound of the range of values to be sampled (hours).

C.5 Triangular

The triangular requires the following three parameters.

Minimum

The lower bound of the range of values to be sampled (hours).

Most Likely

The most likely value must fall between the minimum and maximum values. Its units are hours.

Maximum

The upper bound of the range of values to be sampled (hours).

C.6 Normal

The normal distribution requires two parameters.

Mean

This is the average value in hours of the range to be sampled. See any introductory statistics text for more information on the normal distribution.

Standard Deviation

The standard deviation is a measure of the spread of the distribution. The units in this case are hours. See any introductory statistics text for more information on calculating the standard deviation.

C.7 Weibull

There are several different published forms for the Weibull distribution [Tobias and Trindade, 1995]. The form used in the Consequence Engine for cumulative probability is given in the following equation.

$$P(T <= t) = 1 - e^{-\left(\frac{(t-\theta)}{\alpha}\right)^{\beta}} \qquad T > \theta$$
$$= 0 \qquad \textit{Otherwise}$$

Shape Parameter (β)
This parameter defines the shape of the distribution (dimensionless).

Scale Parameter (α)
This parameter determines the scale of the distribution (hours).

Location Parameter (θ)
This parameter locates the distribution on the time scale (hours).

Appendix D. Schedule Module

The following sections describe the elements of the planned schedule.

D.1 Special Periods

Special periods allow the user to identify periods, other than scheduled maintenance, where the equipment is scheduled to be in a state other than its default state. For example, during a military mission, there may be an interval in which equipment is expected to operate at a higher rate of use (higher speed, rougher terrain, etc.) than its normal operational state. During such intervals, equipment components may age more quickly than normal or random failure rates may increase. Special periods allow such intervals to be included in a simulation scenario.

Start Date
The date and time at which the special period begins.

End Date
The date and time at which the special period ends.

Equipment State
The scheduled state of the equipment for the special period. The equipment state can include special operational states such as higher or lower intensity use. See Section D.2 for a discussion of operational states.

Event Group
Special periods may be grouped together for purposes of defining a simulation scenario. For example, equipment such as an aircraft may be subjected to several intervals of use that are different from each other but also different from the default use represented by normal operation. A sequence of different but stressful maneuvers might be an example. This property is used to tie a group of related intervals together.

Can Be Delayed
This property determines whether this special period (and any others in its group) can be delayed if necessary. For example, a sequence of stressful intervals is scheduled to begin at say 10 AM but maintenance is not yet complete. If this property is set to True, the special period , and any others in its group may be moved to a later time in the schedule.

Delay Time
This property determines the maximum time that this special period may be delayed before it is simply removed from the schedule.

Perform During States
This bitwise-summed parameter indicates all system states during which this special period can occur. See Appendix B for an enumeration of possible system states.

Event Cost
If this special period incurs a cost, this property should reflect that cost.

D.2 Preventive Maintenance

Setting up the preventive maintenance (PM) schedule is more complicated than setting up the special periods schedule. Each PM identifies the failure modes that it addresses and is performed on a specified schedule. The PM schedule may be based on elapsed time

(e.g., every quarter) or on equipment usage (every 1000 hours of use). The time required to perform the PM can be either fixed or variable (specified by a probability distribution). The Consequence Engine may combine PMs when they are scheduled close together. The following inputs determine whether and how PMs are combined:

Combine Time Interval
The combine interval, together with the combine interval units, determines whether two PMs will be combined. For example, if the combine interval is 3 and the combine interval unit is 0 (days), then two PMs will be combined if they are scheduled within 3 days of each other.

Combine Interval Time Units
The time units to be applied to the *combine interval*. The enumerated values for time units are given in Appendix B.

The following information is needed for each PM:

ID
A string that uniquely identifies the current preventive maintenance.

Start Date
The date within the simulation schedule at which this PM is first performed. This property is only used if the PM schedule is based on elapsed time.

Start Age
The component age at which this PM is first performed. This property is only used if the PM schedule is based on usage time. Because of this property, a usage-based PM must specified for each component whereas PMs based on elapsed time can group several component failure modes into a single PM.

PM Interval
A value that specifies how often the PM is performed.

System State
The system state while the PM is underway. This is normally scheduled downtime. Possible states are listed in Appendix B.

Perform During States
This is a summed value representing all the system states during which the PM can be performed. For example, if the PM can be scheduled while the system is operational (state = 1), shut down (state = 64), or non-scheduled (state = 128), the appropriate value is the sum of these or 193. If *operational* is one of the states during which this PM can be scheduled, it does not mean that the system remains operational while the PM is performed. Rather, it means that system operation can be interrupted to perform this PM.

Can Be Delayed
This property determines whether this PM can be delayed if necessary. For example, this PM is normally performed every 1000 hours of operation but when the PM is due, the system is in a special operational state that cannot be interrupted. If this property is set to True, the PM may be moved to a later time in the schedule.

Delay Time
This property determines the maximum time that this PM may be delayed before it is simply removed from the schedule.

Cost
This the cost incurred when this PM is performed. This does not necessarily include parts or labor cost which can be included with input on the failure modes that the PM addresses.

Downtime
This is the time required to perform the PM. To allow for uncertainty, the user can specify a distribution (see Appendix C for the available distribution types and their parameters).

D.3 Failure Modes Addressed by the PM

The following information is required to describe each failure mode addressed by a PM.

Failure Mode ID
The ID of a failure mode being addressed by the PM. This ID must match the ID of a failure mode in the model input (see Section 2.1.4).

Decision
This enumerated value determines the basis for performing the maintenance. Options are *none*, *inspect*, and *test*. If the decision is *none*, the maintenance is performed each time without regard for condition of the component being maintained. If the decision is to *inspect*, the component failure mode is examined to see if it is likely to fail before the next scheduled maintenance. The examination is performed by sampling randomly from the time-to-failure distribution. If the sampled time-to-failure is less than the time the next scheduled maintenance, the repair or replacement is performed. If the sampled time-to-failure is greater than the time to the next scheduled maintenance, maintenance is not performed. If the decision is to test, the component is examined to see if it has failed. If so, it is repaired. See Appendix B for more information on these decision options.

Action
This enumerated value determines which of two possible actions is taken - repair or replace. Repair means that the failure mode is not fully renewed and replace means that it is fully renewed. See Appendix B for details.

Test Cost
The cost of performing any test associated with this failure mode. This input is only used if the *decision* is set to test.

Repair Cost
The cost to repair the failure mode or replace the associated part.

Renewal Fraction
If the chosen *action* is repair, this value specifies the renewal fraction for the failure mode.

Probability of False Positive
If the *decision* choice is to inspect, this provides the probability that the inspection will result in a false positive

Probability of False Negative

If the *decision* choice is to inspect, this provides the probability that the inspection will result in a false negative.

The Consequence Engine maintains three collections that, in addition to equipment failures and repairs, result in the actual chronology of events that the simulation creates. These collections are:

1. Special periods. Special periods are simply read into a collection and sorted chronologically.
2. Scheduled maintenance. Preventive maintenances that are based on elapsed time are used to create a collection of scheduled PM events. All PM events that should occur during the simulation are placed in a collection, sorted chronologically, and those that are scheduled within a combine time interval of each other are combined.
3. Usage-based PMs are simply maintained in a collection. No attempt is made to estimate when they might occur during the simulation time since they are based on a component's cumulative age.

Appendix E. Reliability Module

E.1 Failure Modes

The failure modes that make up the reliability model are characterized by the following information:

Failure Mode ID

A string that uniquely identifies the failure mode.

Failure Mode Name

A short, but more descriptive name for the failure mode. Both a name and an ID are used because many reliability analysts use a hierarchical code to identify the failure mode. Hierarchical codes are helpful in relating the failure mode to a specific component in the system being analyzed. However, such codes can be somewhat cryptic so the name provides a short but more descriptive identifier for the failure mode.

Description

The description is optional but can be used to provide additional information about the failure mode.

Age

The current age of the component failure mode. This must be initialized to the age of the component failure mode at the start of the simulation.

Time In State

The time that the failure mode has spent in the current state. This property is not currently used.

Relative Importance

A value that indicates the relative importance of having this failure mode occur. These values are used to rank the occurrence of different failure modes.

Cost

This is the cost of having this failure mode occur. The cost might include parts and labor necessary to repair the failure.

Failure Mode State

The current state of the failure mode. See Appendix B for failure mode state enumerations.

Nominal Downtime

The nominal amount of time required to return this failure mode to its up state when a failure occurs. This value is used when no time-to-repair distribution is provided.

Age During States

This is a summed value representing all the system states during which this failure mode ages. For example, if the failure mode ages during Operational (state = 1), Operational1 (state = 2), Operational2 (state = 4), and Other Up Time (state = 8), the value is the sum of these or 15.

Repair During States
This is a summed value representing all the system states during which this failure mode can be repaired. For example, if the only time maintenance personnel are not available is when the system is shut down, the value would be the sum of all system state values except system shut down. Not currently used.

Detect During States
This is a summed value representing all the system states during which this failure mode can be detected. Not currently used.

Relative Utilization
Not all failure modes age at the same rate. For example, a particular component might be utilized only half the time that the system is operational. If the time-to-failure distribution is based on full-time utilization, the relative utilization provides a means to account for partial utilization.

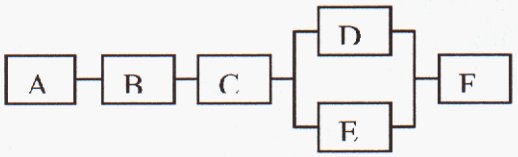
Initial State
The state of the failure mode when the simulation starts.

Time-To-Repair Distribution
The distribution that will be sampled to determine the time-to-repair when a failure occurs. See Appendix C for available distribution types.

Time-To-Failure Distribution
The distribution that will be sampled to determine the probability that the failure mode occurs in a given time interval. See Appendix C for available distribution types.

E.2 Success Paths

A success path is a collection of elements (failure modes, components or subsystems) that, if all are operating, determine the operational state of the system. For example, consider the following simple block diagram model.



Simple Block Diagram Model

In the figure, elements A, B, C and F are in series while D and E are in parallel. If the functionality of the system is unaffected by whether D or E or both are operating, then two success paths would be needed to characterize the system. They are ABCDF and ABCEF, both of which support full functionality. On the other hand, if the functionality of the system is reduced by the failure of either D or E, then three success paths are needed. Success path ABCDEF supports full functionality while ABCDF and ABCEF support reduced functionality. Of course, series elements do not need to be included in any specific success path since they are, by definition, included in all success paths. Success paths are defined by a collection of references to failure modes in the reliability model and a reference to the operational state that results from the success path.

Failure Mode IDs
This is a collection of references to failure modes in the reliability model. Each failure mode ID refers to a member of the success path.
Operational State ID
This string identifies the operational state that results when this success path is active.

E.3 Operational States

The user is able to specify operational states that may differ from the default operational state in terms of its effect on the system being simulated. For example, a helicopter maneuvering at low altitude is subject to more stress than one flying straight and level at higher altitude. During intervals of increased (or decreased) operational stress, selected components and subsystems may effectively age more or less rapidly than normal. Alternate operational states provide a means to treat such effects.
Operational states are characterized by the following properties:
ID
A string that uniquely identifies the operational state.

Affected Failure Modes
This is a collection of failure modes that are affected by the alternate operational state. Each of these affected failure modes includes the following information.
Failure Mode ID
This identifies a failure mode in the reliability model's collection of failure modes.
Acceleration Factor
This factor, which must be greater than 0, causes the failure mode to age more or less rapidly than normal during the alternate operational state.

Appendix F. Spares Module

The spares module treats the spares inventory that is available to the system being simulated. If a failure mode fails during the simulation, the spares inventory is queried to see if a spare part that fixes that component exists, and if it does, how long it will to fix it. The spares inventory is a collection of spare parts each of which has the following properties:

ID
This property uniquely identifies the spare.
Location
The location is used for information only. The effect of a spares location should be reflected in its withdrawal time, restock time, etc.
Restock Time
This is the time required to restock this spare part in the inventory when additional spares are needed.
Emergency Order Time
This is the time required to obtain a spare for the inventory when there are none in stock and a spare is needed quickly.
Purchase Cost
This is the cost to purchase the spare part.
Lot Size
The lot size is the number of this spare that is purchased in each order. This defaults to 1.
Storage Cost
This is the annual cost, per unit, to store this spare. Storage cost should include the actual cost for storage space and may also include depreciation or time value of money considerations.
Shipping Cost
This is the cost to ship the part (or a lot size) when restocking the spare. The shipping cost is added to the purchase cost.
Emergency Shipping Cost
This is the cost to ship the spare (or lot of spares) when an emergency order must be placed.
Reorder Level
When the number of spares in the inventory falls to or below this level, an order is placed for additional spares.
Usage Rate
Because a realistic spares inventory usually supports multiple systems, this property allows the simulation to account for draws from the parts inventory for systems not being simulated directly. The usage rate should represent the rate at which systems, other than the one being simulated, withdraw spares from the inventory.
Withdraw Time
This is the time required to withdraw a spare from the inventory when the spare is in stock. This time is typically added to the system downtime when a part fails and must be replaced.

Appendix G. Cost Module

The cost module assumes that the cost of downtime can be characterized by a function that is piecewise constant. The downtime cost function is characterized by a start date, an end date, and the downtime cost per hour. The cost module requires the following information:

Start Date
The date and time at which the current downtime cost begins.

End Date
The date and time at which the current downtime cost ends.

Downtime Cost
The cost per hour for equipment downtime in the current time interval.

In addition to downtime costs, each event (scheduled or unscheduled) can incur a cost. Each failure mode has an optional cost property. Each scheduled maintenance includes a cost to perform the maintenance, which is added to the cost to repair any failure modes addressed by the maintenance.

Appendix H. Input Description for the Consequence Engine

The Consequence Engine currently reads input data from a text file and does not provide any on-screen input editing capability. However, a utility program has been developed to allow the Consequence Engine to be used for uncertainty calculations and optimization analysis. The utility program makes it relatively easy to use the Consequence Engine for uncertainty analysis using sampled input data from Sandia's SUNS software or to couple the Consequence Engine to the GO optimization driver. This section describes the Consequence Engine input and the procedures for using the Consequence Engine in uncertainty and optimization analysis.

H.1 The Consequence Engine Uncertainty and Optimization Utility

The Consequence Engine uncertainty and optimization utility program allows the Consequence Engine to perform multiple simulations for uncertainty or optimization analysis. The main form for the utility is shown in Figure H.1 below.

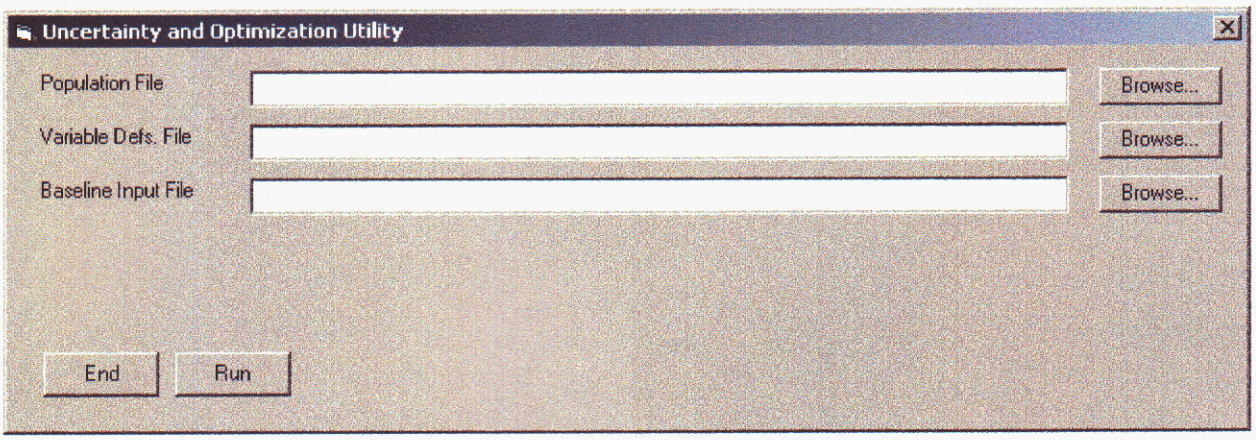


Figure H.1 The Consequence Engine Uncertainty and Optimization Utility Main Form

The utility requires two input files for uncertainty analysis and three input files for optimization. The first file (labeled Population File) is contains either 1) a set of input vectors representing statistically sampled input values for uncertainty analysis or 2) a population of chromosomes representing the next generation of input values for an optimization analysis.

The second file (labeled Variable Defs. File) is only needed for optimization analysis. When the utility is used for uncertainty analysis, the sampled input values represent actual data values that can be used directly and no variable definitions are needed. However, when the utility is used for optimization analysis, the input values in the population file are in the form of integer gene levels that must be interpreted into actual input variable values. The Variable Definitions File provides this interpretation.

The third input file (labeled Baseline Input File) contains the baseline CE input data set. This baseline data set represents the system simulation input prior to any input changes imposed by the uncertainty or optimization analysis.

The form shown in Figure H.1 allows the Consequence Engine to analyze a single set of sampled input vectors (for uncertainty analysis) or a single generation of input values for optimization analysis. Because optimization analysis generally required several runs as repeated generations of input values are created, the utility can also retrieve the file names from a command line. When the utility is run from a command line, the command line should contain the three file names (with paths) separated by the forward slash (/). The order for the files in the command line is 1) population (or input vectors) file, 2) the baseline data set, and 3) the variable definitions file.

The contents of these three files are described in the next three sections.

H.1.1 Population File

The contents of this file depend upon whether uncertainty or optimization analysis is being performed. For uncertainty analysis, the file should contain sampled input values such as provided by Sandia's SUNS sensitivity and uncertainty analysis program. For optimization analysis, the file should contain a population of chromosomes such as provided by Sandia's GO optimization driver.

Uncertainty Analysis

If the utility is used for performing uncertainty analysis, the population file (vectors file) should have a ".vec" extension. The input in the file is comma separated and should be as follows:

Line 1

Number of Input Vectors

This is an integer that indicates the sample size. The actual data set (described below) will contain this many lines.

Number of Input Variables

This integer is the number of CE input variables that will be varied in the uncertainty analysis. The actual data set (described below) will contain this many columns as input to the uncertainty analysis.

Number of Output Variables

This is the number of outputs that the CE will calculate. There is a specific set of outputs that the CE can currently calculate for uncertainty and optimization analysis. These are discussed below. After the utility program is used with the CE for uncertainty analysis, the VEC file will contain the original input vectors with the calculated outputs appended to the end of every vector.

Line 2

Input Variable Names

The second line contains names for each input variable in the order that their sampled values will appear in the input vectors. There are *Number of Input Variables* names separated by commas. The naming convention for inputs that can be varies in uncertainty or optimization analysis is illustrated in Table H.1. For example, SP-RO-PTO Shaft is the reorder level for spare PTO shafts.

Table H.1. Uncertainty and Optimization Variable Naming Convention

Input Category	Code	Input Variable	Code	Full Code
Spares	SP	Reorder Level	RO	SP-RO
Spares	SP	Restock Time	RT	SP-RT
Spares	SP	Withdrawal Time	WT	SP-WT
Use-Based Maintenance	UM	Interval Between Inspections or Maintenance	SI	UM-SI
Use-Based Maintenance	UM	False Positive Probability	FP	UM-FP
Use-Based Maintenance	UM	False Negative Probability	FP	UM-FN

Output Variable Names

The list of input variables is followed by Number of Output Variables names of output variables on the same line of input. Currently available output variables are; 1) Annual Cost, 2) Annual Scheduled Downtime, 3) Annual Unscheduled Downtime, and 4)Annual Total Downtime.

Lines 3 …

Following line 2 which provides names for all input and output variables, there are *Number of Input Vectors* lines where each line represents one input vector – that is, each line contains a sampled value for every input variable in the same order as the input variable names in Line 2.

When the uncertainty analysis is complete, each input vector will have appended to it the calculated values for the four output variables described above (Line 2).

Optimization Analysis

If the utility is used for performing optimization analysis, the population file should not have a ".vec" extension. The input in the file is comma separated and should be as follows:

Line 1

Number of Population Members

This is an integer that indicates number of population members (input data sets) to be analyzed.

Number of Input Variables
This integer is the number of CE input variables that will be varied in the optimization analysis. The actual population members (described below) will contain this many columns as input to the uncertainty analysis.
Number of Output Performance Measures
This is the number of outputs that CE will calculate. There is a specific set of outputs that the CE can currently calculate for uncertainty and optimization analysis. These are discussed below. After the utility program is used with the CE for optimization analysis, the population file will contain the original population members with the calculated performance measures and constraints appended to the end of every population member.
Number of Output Constraints
This is the number of constraints that the CE will calculate. In the current version, no constraints are calculated so this entry should be 0.

Line 2
Input Variable Names
The second line contains names for each input variable in the order that their sampled values will appear in the input vectors. There are *Number of Input Variables* names separated by commas. The naming convention for inputs that can be varies in uncertainty or optimization analysis is illustrated in Table H.1.
Fitness
This entry on line 2 is simply the word "Fitness". This is required by the GO software and is used in case the application that calculated performance measures and constraints also calculates a fitness value.
Performance Measure Names
The next entries on line 2 should be *Number of Output Performance Measures* names of performance measures. Currently available output variables are; 1) Annual Cost, 2) Annual Scheduled Downtime, 3) Annual Unscheduled Downtime, and 4) Annual Total Downtime.
Constraint Names
Since no constraints are currently calculated, no names are required.

Lines 3 ...
Following line 2 which provides names for all input and output variables, there are *Number of Population Member* lines where each line represents one set of input modifications corresponding to a desired optimization run. Each line will contain *Number of Input Variables* integers, comma separated, each representing a gene level setting for the corresponding input variable. Following these integers will be *Number of Output Performance Measures* + 1 real values. On entry to the simulation utility, these will normally be 0. On return, the performance measure values will be the calculated results for each simulation.

H.1.2 Variable Definitions File
The variable definitions file is only needed for an optimization analysis. This file provides the definitions, for each input variable, of the actual variable values corresponding to the gene levels.

Line 1

Number of Options

This is an integer that indicates number of options (genes) in the variable definitions file.

Number of Costs

This integer is the number of costs or constraints associated with each option or gene. In the current version of the CE, this should be 0.

The following lines are repeated *Number of Options* times.

Line 2

Option Name

This string uniquely identifies the option (gene).

Number of Variables in This Option

A given gene may influence multiple variables. This specified the number of variables modified by this gene. In the current version, this should be 1.

Number of Gene Levels

The number of levels or values this gene can assume. For each gene level, each variable affected by the gene must have a specific value.

The following lines are repeated *Number of Variables in This Option* times.

Line 3

Variable Name

This string identifies an input variable to be modified by this option.

Lines 4 through Number of Gene Levels + 3

The next *Number of Gene Levels* lines give values for *Variable Name* – one value for each gene level.

H.1.3 Baseline Input File

This file contains the baseline Consequence Engine simulation input data set. "Baseline" means that the data set describes the simulation input without any of the changes that will be imposed by variable changes for uncertainty analysis or modification options for optimization analysis.

Schedule Input

The [Schedule] segment currently contains only one input value, which is the system default state. See Appendix B for an enumeration of system states.

Special Periods

The [Special Periods] segment describes scheduled intervals in which the system may be in a state other than the default state.

Line 1

Number of Special Period
This integer is the number of special periods in the input file.

Lines 2…
One line of input containing the following information is required for each special period.

Start Date (Date)
The date at which the current special period begins. Note that the date format surrounds the date by pound signs such as #06/19/02#.

End Date (Date)
The date at which the current special period ends. Note that the date format surrounds the date by pound signs such as #06/19/02#.

Equipment State (Integer)
The scheduled state of the equipment for the special period. See Appendix B for enumerated values for the equipment state.

Operating State Reference
This string references a scheduled system operational state that will be active during this special period. If no special operational state is involved, this entry should be blank. Operational state input is described below.

Event Group
This string identifies the special period as belonging to a group of scheduled intervals. The significance of event groups is that if the special period needs to be delayed or canceled, the entire group (all special periods with the same event group identifier) will be delayed or cancelled.

Can Be Delayed
This is a Boolean variable that specified whether the special period is allowed to be delayed in case it cannot occur at the time specified.

Delay
The maximum amount of time that the special period is allowed to be delayed if delays are allowed.

Delay Units
The time units to be applied to Delay. See Appendix B for time unit enumeration.

Event Cost
If this special period incurs a cost, this real value specifies the amount

Perform During States
This bitwise-summed value determines system states that can be interrupted for this special period to take place. If the system is not in one of these states when this period is scheduled, the special period (and all others in the same group) will be delayed up to the maximum Delay specified above. If it still cannot occur, the special period, and all other special periods in the same group will be cancelled.

Scheduled Maintenance

The [Scheduled Maintenance] segment describes preventive maintenance and inspections that are scheduled based on elapsed (calendar) time rather than on usage time.

Line 1

Combine Time Interval (Single)

This input specifies the time interval to be used for combining preventive maintenances.

Combine Interval Time Units (Integer)

This input determines the time units to be applied to the combine time interval. See Appendix B for enumerated values of acceptable time units.

Line 2

Number of Preventive Maintenances (Integer)

This input specifies how many preventive maintenance procedures will be read in. The following lines of input data are repeated for each preventive maintenance.

Line 3

ID (String)

The ID uniquely identifies the preventive maintenance procedure.

Start Date (Date)

This input sets the date within the simulation schedule when this PM will be performed for the first time. Note that the date format surrounds the date by pound signs such as #06/19/02#.

Time Units for PM Interval (Integer)

This input specifies time units that will be applied to the PM interval. See Appendix B for enumerated values of time units.

PM Interval (Single)

The PM will be performed on this interval.

Time Base

For scheduled maintenance, this value should be 1.

Nominal Downtime (Single)

This is the downtime that will be incurred by this PM if the downtime distribution is not sampled. The units are hours.

Cost (Single)

The cost is incurred every time the PM is performed. The cost does not necessarily include the parts and labor cost associated with a specified repair or replacement as those costs can be included in failure mode costs.

System State (Integer)

This is the system state when the PM is being performed. Normally this is scheduled downtime (system state 16). See Appendix B for an enumeration of system states.

Perform During States (Integer)

This is a summed value that represents all the system states in which this PM may be scheduled. Identify all such states from the enumeration of system states in Appendix B then add their values.

Operating State Reference
This string references a scheduled system operational state that will be active during this special period. If no special operational state is involved, this entry should be blank. Operational state input is described below.
Can Be Delayed
This is a Boolean variable that specified whether the special period is allowed to be delayed in case it cannot occur at the time specified.

Delay
The maximum amount of time that the special period is allowed to be delayed if delays are allowed.
Delay Units
The time units to be applied to Delay. See Appendix B for time unit enumeration.

Line 4
Downtime Distribution Type (Integer)
This identifies the type of downtime distribution to be used for this PM. See Appendix C for available downtime distributions.

Number of Downtime Distribution Parameters (Integer)
The number of parameters required by the specified downtime distribution. See Appendix C for distribution details.

Line 5
Parameter Values (Single)
On this line enter *Number of Downtime Distribution Parameters* values. See Appendix C for the parameters that define the available distributions.

Line 10
Number of Failure Modes Addressed by this PM (Integer)
This input determines the number of failure modes to be addressed by this PM.

Line 11…
This line of input must be repeated for each failure mode addressed by this PM.

Failure Mode ID (String)
The ID of the failure mode addressed by this PM. This ID must match one of the failure mode ID's in the Model segment (see below).

Decision (Integer)
This enumerated value determines the basis for performing the maintenance. Options are *none (0)*, *inspect (1)*, and *test (2)*. See Appendix B for more information on these decision options.

Action (Integer)
This enumerated value determines which of two possible actions is taken – repair (0) or replace.(1) See Appendix B for details.

Test Cost (Single)
The cost of performing any test associated with this failure mode. This input is only used if the *decision* is set to test.

Repair Cost (Single)
Enter the cost to repair the failure mode or replace the associated part.

Renewal Fraction (Single)
If the chosen *action* is repair, this value specifies the renewal fraction for the failure mode. (This input is not yet used.)

Probability of False Positive (Single)
If the decision choice is to test, this provides the probability that the test will result in a false positive. (This input is not yet used.)

Probability of False Negative (Single)
If the decision choice is to test, this provides the probability that the test will result in a false negative. (This input is not yet used.)

Lines 3 through 11… must be repeated for every PM.

Use-Based Maintenance

The [Use-Based Maintenance] segment describes preventive maintenance and inspections that are scheduled based on equipment usage time.

Line 1
Number of Use-Based Maintenances (Integer)
This input specifies how many preventive maintenance procedures will be read in. The following lines of input data are repeated for each preventive maintenance.

Line 3
ID (String)
The ID uniquely identifies the preventive maintenance procedure.

Start Age (Single)
This input sets the component age when this PM will be performed for the first time. After the first occurrence of this PM, the maintenance will be repeated according to the value of the PM interval (below).

Time Units for PM Interval (Integer)
This input specifies time units that will be applied to the PM interval. See Appendix B for enumerated values of time units.

PM Interval (Single)
The PM will be performed on this interval measured in usage time of the component.

Time Base
For scheduled maintenance, this value should be 2.

Nominal Downtime (Single)
This is the downtime that will be incurred by this PM if the downtime distribution is not sampled. The units are hours.

Cost (Single)
The cost is incurred every time the PM is performed. The cost does not necessarily include the parts and labor cost associated with a specified repair or replacement as those costs can be included in failure mode costs.

System State (Integer)
This is the system state when the PM is being performed. Normally this is scheduled downtime (system state 16). See Appendix B for an enumeration of system states.

Perform During States (Integer)
This is a summed value that represents all the system states in which this PM may be scheduled. Identify all such states from the enumeration of system states in Appendix B then add their values.

Operating State Reference
This string references a scheduled system operational state that will be active during this special period. If no special operational state is involved, this entry should be blank. Operational state input is described below.
Can Be Delayed
This is a Boolean variable that specified whether the special period is allowed to be delayed in case it cannot occur at the time specified.

Delay
The maximum amount of time that the special period is allowed to be delayed if delays are allowed.
Delay Units
The time units to be applied to Delay. See Appendix B for time unit enumeration.

Line 4
Downtime Distribution Type (Integer)
This identifies the type of downtime distribution to be used for this PM. See Appendix C for available downtime distributions.

Number of Downtime Distribution Parameters (Integer)
The number of parameters required by the specified downtime distribution. See Appendix C for distribution details.

Line 5
Parameter Values (Single)
On this line enter *Number of Downtime Distribution Parameters* values. See Appendix C for the parameters that define the available distributions.

Line 10
Number of Failure Modes Addressed by this PM (Integer)
This input determines the number of failure modes to be addressed by this PM.

Line 11…
This line of input must be repeated for each failure mode addressed by this PM.

Failure Mode ID (String)
The ID of the failure mode addressed by this PM. This ID must match one of the failure mode ID's in the Model segment (see below).

Decision (Integer)
This enumerated value determines the basis for performing the maintenance. Options are *none (0)*, *inspect (1)*, and *test (2)*. See Appendix B for more information on these decision options.

Action (Integer)
This enumerated value determines which of two possible actions is taken – repair (0) or replace.(1) See Appendix B for details.

Test Cost (Single)
The cost of performing any test associated with this failure mode. This input is only used if the *decision* is set to test.

Repair Cost (Single)
Enter the cost to repair the failure mode or replace the associated part.

Renewal Fraction (Single)
If the chosen *action* is repair, this value specifies the renewal fraction for the failure mode. (This input is not yet used.)

Probability of False Positive (Single)
If the decision choice is to test, this provides the probability that the test will result in a false positive. (This input is not yet used.)

Probability of False Negative (Single)
If the decision choice is to test, this provides the probability that the test will result in a false negative. (This input is not yet used.)

Lines 3 through 11… must be repeated for every PM.

Normal Operational State

The [Normal Operation] segment provides information on the normal operational state of the system (See Appendix B for system state enumeration). The purpose of operational state definitions is to allow different operational states to stress failure modes differently and to permit the system to perform at different levels. For example, the airframe of a military aircraft may age more rapidly during periods of high aerodynamic stress than when the plane is flying straight and level.

Line 1
Number of Operational States
For the Normal Operation segment, this should be 1.

Line 2
Operational State ID
This is a string that uniquely identifies this operational state. In this segment, the ID should be "Normal Operation."
Number of Affected Failure Modes
This integer is the number of failure modes that will age more or less rapidly during this operational state. For normal operation, this should be 0.

Line 3

Number of Condition Performance Measures

This integer is the number of condition performance measures. As this feature is not yet implemented, this entry should be 0.

Line 4

Number of Quantity Performance Measures

This integer is the number of quantity performance measures. As this feature is not yet implemented, this entry should be 0.

Operational State 1

The [Operational1] segment provides information on the operational state of the system other than the normal operational state (See Appendix B for system state enumeration). The purpose of operational state definitions is to allow different operational states to stress failure modes differently and to permit the system to perform at different levels. For example, the airframe of a military aircraft may age more rapidly during periods of high aerodynamic stress than when the plane is flying straight and level. Operational State 1must be scheduled and failure mode acceleration factors can be specified for each scheduled occurrence.

Line 1

Number of Operational1 States

This integer is the number of different Operational1 states to be used in the simulation.

The following lines must be repeated for every Operational1 state.

Line 2

Operational1 State ID

This is a string that uniquely identifies this Operational1 state.

Number of Affected Failure Modes

This integer is the number of failure modes that will age more or less rapidly during this operational state.

The following lines must be repeated for every failure mode affected by this operational state.

Line 3

Failure Mode ID

This string identifies a failure mode in the reliability model.

Acceleration Factor

This real value is applied as an aging acceleration factor and must be positive. When the system is in this operational state, the referenced failure mode will age at a rate that is normal elapsed time multiplied by this factor.

The information in line 3 must be repeated for every failure mode that is affected by this operational state. Following input on the affected failure modes, there should be two lines, each with 0 as the only input.

Operational State 2

The [Operational2] segment provides information on operational states of the system that differ from the normal operational state (See Appendix B for system state enumeration). The Operational2 state cannot be scheduled but can be transitioned to.

Line 1

Number of Operational2 States

This integer is the number of different Operational1 states to be used in the simulation.

The following lines must be repeated for every Operational2 state.

Line 2

Operational2 State ID

This is a string that uniquely identifies this Operational2 state.

Number of Affected Failure Modes

This integer is the number of failure modes that will age more or less rapidly during this operational state.

The following lines must be repeated for every failure mode affected by this operational state.

Line 3

Failure Mode ID

This string identifies a failure mode in the reliability model.

Acceleration Factor

This real value is applied as an aging acceleration factor and must be positive. When the system is in this operational state, the referenced failure mode will age at a rate that is normal elapsed time multiplied by this factor.

The information in line 3 must be repeated for every failure mode that is affected by this operational state. Following input on the affected failure modes, there should be two lines, each with 0 as the only input.

Success Paths

The [Success Paths] segment provides the information required to determine whether the system is operational at a given point in time and, if so, the applicable operational state.

Line 1

Number of Series Failure Modes

This is the number of failure modes that are in series in the reliability model. Series failure modes, if they occur, cause the system to fail.

Number of Success Paths

This integer is the number of success paths to be read.

Line 2 to N

Failure Mode ID

This string is the ID for a failure mode in the reliability model. There should be *Number of Series Failure Modes* lines of these Failure Mode Ids.

The following lines are repeated *Number of Success Path times*.

Line N + 1

Operational State Reference

This string identifies the operational state (Normal Operation or Operational2) that the system will be in when this success path is operative.

Number of Failure Modes

This integer is the number of failure modes in this success path.

Lines N+2 to M

Failure Mode ID

This string references a failure mode in the reliability model. This collection of *Number of Failure Modes* strings makes up a success path such that, if all series failure modes and all failure modes in this success path are true (not failed), then the system is in the referenced operational state. The success paths should be entered in hierarchical order with the highest operational state first.

Reliability Model

The reliability model segment ([Model]) defines the reliability model used as a basis for the simulation.

Line 1

Model Name (String)

Enter a descriptive string for the reliability model. This input is not currently used but will be used as an output label in later versions.

Start Date (Date)

This is the start date for the simulation. Note that the date format surrounds the date by pound signs such as #06/19/02#.

Simulation Duration (Single)

The simulation will be performed for this time period where the time units are defined below.

Number of Simulations (Integer)

This is the number of times the simulation will be repeated.

Duration Time Units (Integer)

The time units to be applied when determining the simulation duration. Time unit options are enumerated in Appendix B.

Description (String)
This input describes the simulation. While this input is not currently used, it will be used to label output results in a later version.

Randomize Initial Age (Boolean)
This input determines whether the initial ages of the failure modes is the same in each simulation or is randomized. If you want to perform repeated simulations to get adequate statistics on a specific simulation scenario, you may want to set this value to #FALSE# so that the same scenario is repeated each time. On the other hand, if you want to simulate a long time period, it may be faster to run several simulations for a shorter duration and set this value to #TRUE#.

Randomize Seed (Boolean)
If set to #TRUE#, a new seed will be selected randomly for each simulation.

Line 2

Number of Failure Modes (Integer)
This determines the number of failure modes that will be read in.

Line 3

Failure Mode ID (String)
This string uniquely identifies the failure mode. The ID must be different for each failure mode.

Failure Mode Name (String)
This is a short, but more descriptive name for the failure mode. Both a name and an ID are used because many reliability analysts use a hierarchical code to identify the failure mode. Hierarchical codes are helpful in relating the failure mode to a specific component in the system being analyzed. However, such codes can be somewhat cryptic so the name provides a short but more descriptive identifier for the failure mode. A value must be entered but it can be a blank string (that is ""). Failure mode names will be used as graphics labels in a later version.

Description (String)
The description can be used to provide more detail on the failure mode. A value must be entered but it can be a blank string (that is "").

Failure Mode Age (Single)
This is the age (in hours) of the failure mode at the start of the simulation.

Time-In-State (Single)
This is the time (in hours) that the failure mode has been in its initial state (see input below) at the start of the simulation. This would normally be the same as the *Failure Mode Age*.

Relative Importance (Single)
This input indicates the relative importance of having this failure mode occur. Relative importance is used to rank the occurrence of different failure modes.

Nominal Cost (Single)
This input specifies the cost of having this failure mode occur. The cost might include parts and labor necessary to repair the failure.

Line 4

Failure Mode State (Integer)
This is the initial state of the failure mode. Possible failure mode states are Up, Incipient, Failed, and Scheduled Down. See Appendix B for failure mode state enumeration.

Nominal Downtime (Single)
The nominal amount of time required to return this failure mode to its up state when a failure occurs.

Age During States (Integer)
Enter a summed value representing all the system states during which this failure mode ages. See Appendix B for enumeration values of system states.

Repair During States (Integer
Enter a summed value representing all the system states during which this failure mode can be repaired. See Appendix B for enumeration values of system states.

Detect During States (Integer)
This is a summed value representing all the system states during which this failure mode can be detected. See Appendix B for enumeration values of system states. Not currently used.

Relative Utilization (Single)
Enter a value greater than 0 and less than or equal 1 to represent the utilization of this component failure mode relative to the system.

Initial State (Integer)
The state of the failure mode when the simulation starts.

Line 5

Time-To-Repair Distribution Type (Integer)
Specify the type of distribution to be used for the time-to-repair distribution. See Appendix C for available distribution types.

Number of Time-to-Repair Distribution Parameters (Integer)
Indicate the number of distribution parameters to be read. The number of parameter values and their meaning depends on the distribution type. See Appendix C for details.

Line 6

Time-to-Repair Distribution Parameters (Single)
Enter the number of values specified in line 5. The available distributions and the meaning of their parameters can be found in Appendix C.

Line 7

Time-To-Failure Distribution Type(Integer)

Specify the type of distribution to be used for the time-to-failure distribution. See Appendix C for available distribution types.

Number of Time-to-Failure Distribution Parameters (Integer)

Indicate the number of distribution parameters to be read in. The number of parameter values and their meaning depends on the distribution type. See Appendix C for details.

Line 8

Time-to-Failure Distribution Parameters (Single)

Enter the number of values specified in line 5. The available distributions and the meaning of their parameters can be found in Appendix C.

Line 9

Number of Spares

This integer is the number of spares that can be used to repair this failure mode. In the current version, this should be 0 (no spare needed) or 1.

Line 10

Spare ID

The name of the spare needed to repair this failure mode.

Lines 3 through 10 must be repeated for every failure mode in the model.

Spares Input

The [Spares] segment provides the input required to describe any spares inventory to be used in the simulation.

Line 1

Number of Spares

This integer is the number of spares that will be in the inventory.

Line 2

Random Seed

This long integer is a random seed used to initialize the spares inventory model.

Last Update

This date value specifies the last date upon which the spares inventory was updated. This would normally be the start date of the simulation if you assume that the spares inventory was up-to-date at simulation start.

Number of Systems

This integer specifies the number of systems (not including the one being simulated) that the spares inventory will support. This input helps to establish a realistic inventory since it is difficult to create a realistic spares inventory for a single system.

The remaining lines are repeated Number of Spares times.

Lines 3

Spare ID

This string uniquely identifies the spare part.

Location

This string is used for information only and identifies the location of the spare.

Lot Size

This integer determines how many of this spare are ordered at a time.

Restock Time

The time required to receive a normal order for this spare.

Restock Time Units

The time units used to interpret the Restock Time.

Emergency Order Time

The time required to receive an emergency order for this spare. Emergency ordering occurs when a spare is demanded that is not in stock.

Emergency Order Time Units

The time units used to interpret the Emergency Order Time.

Purchase Cost

The cost per unit to purchase this spare

Storage Cost

This is the annual cost, per unit, to store this spare. Storage cost should include the actual cost for storage space and may also include depreciation or time value of money considerations.

Shipping Cost

This is the cost to ship the part (or a lot size) when restocking the spare. The shipping cost is added to the purchase cost.

Emergency Ship Cost

This is the cost to ship the part (or a lot size) when placing an emergency order for the spare. The shipping cost is added to the purchase cost.

Reorder Level

When the number of spares in the inventory falls to or below this level, an order is placed for additional spares.

Units In Stock

This integer is the number of units in stock at the start of the simulation.

Usage Rate

Because a realistic spares inventory usually supports multiple systems, this property allows the simulation to account for draws from the parts inventory for systems not being simulated directly. The usage rate should represent the rate at which systems, other than the one being simulated, withdraw spares from the inventory.

Usage Rate Time Units

This is the unit of time used to interpret the usage rate.

Withdraw Time

This is the time required to withdraw a spare from the inventory when the spare is in stock. This time is typically added to the system downtime when a part fails and must be replaced.

Withdraw Time Units
This is the unit of time used to interpret the withdrawal time.

Line N + 1

Number of Existing Part Orders
This is the number of unfilled part orders at the beginning of the simulation. If the last update date is at the start of the simulation, this should be 0.

The following line should be repeated *Number of Existing Part Orders* times.

Line N+2

Date Order was Placed
This is the date that the order was placed.
Date Due
This is the date the part order is due to be received.
Number Ordered
This is the number of parts ordered.

Cost Function Segment

This segment allows the user to set up a time-dependent cost function to be applied to scheduled and unscheduled downtime.

Line 1

Number of Cost Values (Integer)
Indicate the number of cost and date values to be provided to set up the downtime cost function.

Line 2 ...

Start Date (Date)
Enter the date at which the current cost-per-hour becomes applicable. Note that the date format surrounds the date by pound signs such as #06/19/02#.

Cost per Hour (Single)
Enter the cost of downtime (scheduled or unscheduled) in dollars per hour that is to be applied from the current start date to the start date of the next interval.

Appendix I. Output Description for the Consequence Engine

In the current version of the CE, output results are simply appended to the end of the input file in a segment labeled [Results]. The results are presented in a way that makes it simple to paste them into a spreadsheet for analysis.

I.1 Simulation Summary

This portion of the output summarizes the results by simulation run. It is preceded by a header row that provides column headers if you elect to paste the results into a spreadsheet. The number of rows following the header row is equal the number of simulations. Each row contains the following information:

Simulation Number

The ordinal number of the simulation summarized by this row of output.

Number of Failures

The number of failures that occurred in this simulation.

Operational Time

The total amount of time (hours) that the system spent in the Operational state.

Operational1 Time

The total amount of time (hours) that the system spent in the Operational1 state.

Operational2 Time

The total amount of time (hours) that the system spent in the Operational2 state.

Cost

The total cost incurred by the system during this simulation.

Unscheduled Downtime

The total unscheduled downtime (hours) incurred by the system during this simulation.

Scheduled Downtime

The total scheduled downtime (hours) incurred by the system during this simulation.

Non-Scheduled Time

The total time (hours) that the system spent in the non-scheduled state.

I.2 Number of Failures by Failure Mode

This block of output presents the number of times each failure mode occurred by simulation. The first row provides column headers consisting of failure mode ID's. Each successive row contains the following information:

Simulation Number

The ordinal number of the simulation summarized by this row of output.

Number of Failures

The number of times each failure mode occurred in the current simulation.

I.3 Cost by Failure Mode

This block of output presents the cost incurred for each failure in the simulation. The first row provides column headers consisting of failure mode ID's. Each successive row contains the following information:

Simulation Number

The ordinal number of the simulation summarized by this row of output.

Cost
The cost attributable to each failure mode in the current simulation.

I.4 Unscheduled Downtime by Failure Mode

This block of output presents the unscheduled downtime incurred for each failure in the simulation. The first row provides column headers consisting of failure mode ID's. Each successive row contains the following information:
Simulation Number
The ordinal number of the simulation summarized by this row of output.
Unscheduled Downtime
The unscheduled downtime attributable to each failure mode in the current simulation.

I.5 Other Output

Time-to-failure and cost statistics are available when the CE is used as a component in another application. These results are accessed by instantiating the appropriate objects as follows:

```
Dim oCostStats    As SummaryStats
Dim oTTFStats     As SummaryStats
Set oCostStats = CE.CostStats
Set oTTFStats = CE.TTFStats
```

The SummaryStats class provides the following properties:
Mean
The mean value of the cost or time-to-failure results from repeated simulations.
StdDev
The standard deviation of the cost or time-to-failure results from repeated simulations.
PTile_1
The first percentile of the cost or time-to-failure results from repeated simulations.
PTile_5
The fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_10
The tenth percentile of the cost or time-to-failure results from repeated simulations.
PTile_15
The fifteenth percentile of the cost or time-to-failure results from repeated simulations.
PTile_20
The twentieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_25
The twenty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_30
The thirtieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_35
The thirty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_40
The fortieth percentile of the cost or time-to-failure results from repeated simulations.

PTile_45
The forty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_50
The fiftieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_55
The fifty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_60
The sixtieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_65
The sixty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_70
The seventieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_75
The seventy-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_80
The eightieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_85
The eighty-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_90
The ninetieth percentile of the cost or time-to-failure results from repeated simulations.
PTile_95
The ninety-fifth percentile of the cost or time-to-failure results from repeated simulations.
PTile_99
The ninety-ninth percentile of the cost or time-to-failure results from repeated simulations.

Distribution

Unclassified – Internal Distribution only

1	MS 9018	Central Technical Files 8945-1
2	MS 0899	Technical Library, 9616
1	MS 1176	James E. Campbell, 15312
1	MS 1176	Robert M. Cranwell, 15312
1	MS 1176	Kelly S. Lowder, 15312
1	MS 1170	Russell D. Skocypec, 15310
1	MS 1176	Laura P. Swiler, 9211
1	MS 0323	Donna L. Chavez, LDRD Office, 1010